VMware Project North Star (Thinstall) User's Manual

VMware Project North Star (Thinstall) Beta 1

Beta

vmware®

VMware Project North Star (Thinstall) User's Manual
Revision: 20080324
Item: XXX-ENG-QNNN-NNN

You can find the most up-to-date technical documentation on our Web site at:

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

Beta

© 2008 VMware, Inc. All rights reserved. Protected by one or more U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156, 6,795,966, 6,880,022, 6,944,699, 6,961,806, 6,961,941, 7,069,413, 7,082,598, 7,089,377, 7,111,086, 7,111,145, 7,117,481, 7,149,843, 7,155,558, 7,222,221, 7,260,815, 7,260,820, 7,269,683, 7,275,136, 7,277,998, 7,277,999, 7,278,030, 7,281,102, and 7,290,253; patents pending.

VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book 7

1 Introduction 9
- Features 10
  - User Mode 16
  - Side-by-Side (SxS) 18
  - Dynamic Path Relocation 19
  - Short Pathnames 21
  - Virtual Services 24
- Change Log 26

2 Getting Started 35
- Using a Clean PC for Capturing 41
- Installing VM Server 41

3 How VMware Project North Star (Thinstall) Works 51
- Block based Streaming 53
- Supported OSs & Apps 55
- Desktop Integration 56
- Upgrading applications 57
- Limitations 59

4 Isolation Modes 61
- Understanding Isolation Modes 62

5 Sandbox Overview 65
- Sandbox Structure 67

6 Utilities and Automation Tools 69
- Setup Capture 69
  - Package.ini generation 72
  - Reboot continue 73
- Log Monitor 74
  - Locating Errors 75
  - Log Format 76
  - Forensics Example 80
- sbmerge 85
- ThinReg 86
- snapshot 87
- dll_dump 88
- VREGTool 89
- TLink 91

Beta

7 MSI Generation 93

8 Package.ini format 95

AccessDeniedMsg 97
AddPageExecutePermi... 98
AutoShutdownServices 98
AutoStartServices 98
BlockSize 99
ChildProcessEnvironme... 99
ChildProcessEnvironme... 99
CommandLine 100
CompressionType 100
DirectoryIsolationMode 101
DisableTracing 101
Disabled 102
ExcludePattern 102
ExternalCOMObjects 102
ExternalDLLs 103
Icon 103
InventoryName 104
IsolatedMemoryObjects 104
IsolatedSynchronizatio... 105
LogPath 105
MSIDefaultInstallAllUsers 105
MSIFilename 106
MSIInstallDirectory 106
MSIManufacturer 107
MSIProductCode 107
MSIProductVersion 107
MSIRequireElevatedPri... 108
MSIUpgradeCode 108
NetRelaunch 108
NoRelocation 109
RegistryIsolationMode 109
PermittedGroups 110
ReserveExtraAddress... 110
RetainAllIcons 111
RemoveSandboxOnExit 111
SandboxName 111
SandboxPath 112
SandboxNetworkDrives 112
SandboxRemovableDisk 112
StripVersionInfo 113
Source 113
Shortcut 113
UpgradePath 114
VirtualizeExternalOut... 114
VirtualDrives 114
Version.XXXX 115
WorkingDirectory 116

Beta

9 Scripting 117
Callback functions 117
Examples 118
.bat example 118
timeout example 118
Registry Modify 119
.reg example 119
Stopping Service 119
copyfile example 119
System registry example 120

10 API functions 121
AddForcedVirtualLoadPath 121
ExitProcess 122
ExpandPath 122
ExecuteExternalProcess 123
ExecuteVirtualProcess 123
GetBuildOption 124
GetFileVersionValue 124
GetCommandLine 125
GetCurrentProcessName 125
GetOSVersion 125
GetEnvironmentVariable 126
RemoveSandboxOnExit 127
SetEnvironmentVariable 127
SetFileSystemIsolation 127
SetRegistryIsolation 128
WaitForProcess 128

11 Access Control using Active Directory 131

12 Application PATH and Environment Variables 133

13 Tips and Tricks 135
Packaging .NET Applications 135
Precaptured .NET 135
Reducing size 136
Tips and Tricks for Specific Applications 137
Office 2007 137
Outlook 137
Explore.exe 138
Java Runtime Environment (JRE) 139

14 Snapshot.exe 141

15 Virtual Registry 143
Virtual Registry—Text Format 144
##Attributes.ini 145

16 Virtual Filesystem 147
Folder Macros 148
##Attributes.ini 150

Beta

17 Frequently Asked Questions 151
Application Updates 153
Plugins and Addins 154
Large Packages 155

Index 157

Beta

About This Book

The *VMware Thinstall Installation and Adminstration Manual*, provides information about installing and configuring Thinstall, including how to upgrade your software, configure packages, configure isolation modes and sandboxes, use the utilities and tools, configure MSI files, use scripts, and use License Manager.

Intended Audience

This book is intended for anyone who wants to install, upgrade, or use VMware Thinstall. Typical users will be system administrators responsible for the distribution and maintenance of corporate software packages.

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to:

docfeedback@vmware.com

Technical Support and Education Resources

The following sections describe the technical support resources available to you. You can access the most current versions of this book and other books by going to:

http://www.vmware.com/support/pubs

VMware Education Services

VMware courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. For more information about VMware Education Services, go to http://mylearn1.vmware.com/mgrreg/index.cfm.

Beta

Beta

Introduction 1

The following links are useful to have:

- Chapter 2, "Getting Started," on page 35: A quick walk-thru of how to capture and build virtualized applications
- "Features" on page 10: A list of a few key features in VMware Project North Star (Thinstall)
- Flash walk-thru demos:
  - How to Thinstall an Application (http://thinstall.com/demos/vs_beta_intro/)

    Takes you through the process of virtualizing an application with VMware Project North Star (Thinstall)
  - How to Thinstall IE based Java applets (http://thinstall.com/demos/java_applet)

    Learn how to Thinstall the Java Runtime Environment and the host PC's Internet Explorer to run web applets without installation.
  - How to Thinstall a .NET 2.0 Application (http://thinstall.com/demos/dnet20/)

    Learn how to Thinstall a .NET 2.0 application. See how VMware Project North Star (Thinstall) can be used to package the .NET framework. (10 min)
  - Thinstalling ActivePerl (http://thinstall.com/demos/perl)

    Learn how to Thinstall ActiveState ActivePerl plus a perl script into a single EXE and have it run with a click and no installation.
- http://thinstall.com/thintalk VMware Project North Star (Thinstall) Discussion forum.
  - Find out what apps other users have tested
  - Post your results
  - Request features
  - Ask general questions about VMware Project North Star (Thinstall)
  - Various tips & tricks, direct from the VMware Project North Star (Thinstall) development team

    (you must be logged into your account to access the blog & discussion forum)
- "Change Log" on page 26: What's new in this version

Beta

Features

VMware Project North Star (Thinstall) includes the following features:

100% user-mode

VMware Project North Star (Thinstall) is the only Virtualization solution available which runs completely in user-mode ("User Mode" on page 16). This has many system stability, security, infrastructure, and ease-of-use implications. For more details, see "User Mode" on page 16.

Support for Virtual Side-by-Side (SxS)

VMware Project North Star (Thinstall) is the only Virtualization solution to fully support SxS, a critical feature for deploying most new applications. For more details, see "Side-by-Side (SxS)" on page 18.

No device drivers

Because VMware Project North Star (Thinstall) requires no device drivers, it can run applications without Administrator rights and requires no changes to the PC even if the user is running on a locked-down PC. Microsoft reports 3rd party device drivers cause over 80% of machine crashes, and a large number of new machine-wide vulnerabilities come from bugs in device drivers.

Application Isolation capabilities

VMware Project North Star (Thinstall) allows applications to run without any modification to the host PC's registry or file system. Other applications running on the same PC will not be aware of virtualized applications, so regression testing (a major cost in application deployment) can be eliminated or drastically reduced. Much of the cost for application deployment relates to testing new applications against other applications deployments. Some other Virtualization solutions make registry and file system changes virtual to the entire system temporarily or permanently, so regression testing continues to be needed and application roll-outs still have the possibility for breaking other applications on the desktop.

Multiple simultaneous client versions

VMware Project North Star (Thinstall) is the only virtualization technology which supports multiple concurrent running copies of the client on the same PC. This means you can package Application A using VMware Project North Star (Thinstall) 3.066 and deploy it to 500 user desktops. Later, you may upgrade to VMware Project North Star (Thinstall) 3.500 to take advantage of some new features. You can deploy a new Application B using VMware Project North Star (Thinstall) 3.500 without affecting previously deployed Application A. This is especially critical when multiple divisions in a company want to use the technology independently and do not need to synchronize around a central version. In fact, VMware Project North Star (Thinstall) has been used to deploy application over 100 million desktops around the world through various software developers and publishers. These applications will continue to operate independently of VMware Project North Star (Thinstall). Other Virtualization solutions only support one version of their client on a single PC at any time, and since any application deploying new versions is equivalent to deploying a major service pack to all of your PCs companies can only afford to upgrade once or twice a year at a significant cost. Because of VMware Project North Star's ability to support multiple concurrent client versions, VMware Project North Star (Thinstall) has a very fast product release cycle (once a month) and customers can take advantage of new features immediately without impacting previously deployed applications.

Instant Portable Deployment (USB Flash / CDROM)

VMware Project North Star (Thinstall) can easily convert standard applications like Microsoft into portable applications which run from USB Flash or CDROM. For USB deployment's VMware Project North Star's portable mode redirects application registry and filesystem changes intended for the host PC to files stored on the portable device. Because VMware Project North Star (Thinstall) has no device drivers and runs in Guest / Restricted user accounts, you can use Thinstalled portable applications on kiosk PCs even if they are locked-down and do not permit any installation.

Wide Platform support

- Windows NT, 2k, XP, w2k3, *Vista*

Beta

- Terminal Server & Citrix Metaframe
- Windows PE (Preinstalled Environment)
- 16 and 32bit apps on *64 bit Vista & XP*
- Single package can support all platforms
- Automatic migration for: NT/2k->XP and XP->Vista

Group Policy cannot be circumvented

Because VMware Project North Star (Thinstall) has no kernel-mode code, it cannot violate machine Group Policy applied to user accounts. This makes VMware Project North Star (Thinstall) safe to deploy in environments where security and stability are important.

VMware Project North Star (Thinstall) has no ability to give application elevated permissions to devices on the machine, such as the real file `system/registry`, networking devices, printers, etc. For companies that have invested a lot of time constructing account security policies, they can rest assured that the Microsoft OS team is fully responsible for implementing a secure environment, and VMware Project North Star (Thinstall) does not bring any file system filters which might modify that.

Runs in restricted user accounts

Because VMware Project North Star (Thinstall) requires no device drivers, it can run application in Guest User accounts without any previous install.

Allows apps requiring Admin rights to run without additional privileges

Many applications fail to run without Admin rights because they expect to be able to write to global locations like `HKEY_LOCAL_MACHINE` and `c:\program` files. Using sandbox technology, VMware Project North Star (Thinstall) makes applications believe they have the ability to make global changes when they are actually writing to user+app specific locations. This feature allows applications to run in security restricted environment like Terminal Server and Vista. Using VMware Project North Star (Thinstall) it's possible to take many older applications and convert them to Vista or multi-user applications.

VMware Project North Star (Thinstall) Runtime embeds in packages

Deployment cannot get any simpler and fool-proof than using VMware Project North Star (Thinstall). VMware Project North Star (Thinstall) accomplishes it's zero-footprint and no installation by embedded it's entire runtime in each executable that it packages. Because the VMware Project North Star (Thinstall) runtime is super-small (~400k), and package data is stored in a compressed state, the overall disk footprint is usual 2X smaller than traditional deployments of the same application not using VMware Project North Star (Thinstall).

VMware Project North Star (Thinstall) is loaded by Windows as a normal application

The impact of deploying Thinstalled applications is the same as deploying any normal application, from Window's perspective the EXEs generated by VMware Project North Star (Thinstall) are simple notepad-like applications that run without external dependencies. Other applications on the system are not effected by deploying Thinstalled applications.

VMware Project North Star (Thinstall) is small, light weight, & fast

VMware Project North Star (Thinstall) can be loaded into memory in a few milliseconds (even over network) and quickly loads the application in question. In some cases VMware Project North Star (Thinstall) can actually load applications faster than Windows. The entire VMware Project North Star (Thinstall) runtime occupies ~400k on disk!

Setup Capture

VMware Project North Star's Setup Capture program makes conversion of traditional apps to virtual apps a very simple process. Setup Capture can be used to take two snapshots of a machine before and after an application is installed and generate a VMware Project North Star (Thinstall) project directly from these changes. The process is simple enough that it can be used by IT staff with a wide variety of skill levels. Setup Capture's snapshot process is the fastest in the industry, it can scan a clean version of Windows XP in

Beta

approximately 20 seconds. Setup Capture supports continuation after reboot. Command-line tools allow you to print out differences between two machine snapshots and the snaphost process can be integrated into other tools or as part of your testing/verification process. Setup Capture can be run directly from a network share on clean machine without any installation, enabling you to capture untainted images of machines.

Virtual short path names

VMware Project North Star (Thinstall) is the only solution that sits above the windows loader and file system and the only solution to properly handle short pathnames (DOS 8.3 file names) ("Short Pathnames" on page 21). Other virtualization solutions fail to work when applications are not installed to DOS 8.3 compatible paths and then moved to other computers. For example, applications like Microsoft Office has a large number of registry values that contain entries like `C:\PROGR~1\MICROS~3`, however on other computers the virtual files may actually appear to exist at `C:\PROGRA~1\MICROS~4` and because of this various COM components fail to work at runtime. Because of this, other solution require you to capture applications installed to DOS 8.3 compatible paths like `c:\OFFICE`. The issue is that many applications will not install or run properly when their non-default path is used. VMware Project North Star (Thinstall) elegantly eliminates the need to worry about short paths by using dynamic macro expansion for all registry and filename information. At runtime, VMware Project North Star (Thinstall) will filter registry and filename data to replace short paths with macro versions that re-expand to the correct location on new computers. In the scenario above registry values will automatically re-adjust to point to the correct locations when the package is run on a different computer. For more information, see "Short Pathnames" on page 21.

Easy Package editing & builds

VMware Project North Star (Thinstall) uses a directory based structure to store captured projects, allowing for easy browsing, search, editing, and modification using standard file system tools like explorer and Windows search. There are not complicated new user interfaces to learn. If you know how to browse your file system, you know how to edit VMware Project North Star (Thinstall) projects. Because VMware Project North Star (Thinstall) projects are simply directory with no absolute path information, they can be easily copied from computer to computer, or be hosted and compiled directly from network shares. The VMware Project North Star (Thinstall) build process is similar to a compiler & linker and should be familiar to most IT and development shops.

Text based projects and settings integrate with source control systems

Except for application files, which are stored as normal binaries all VMware Project North Star (Thinstall) files and settings are stored as simple .ini files so VMware Project North Star (Thinstall) projects can be managed using source-control technologies like SourceSafe, CVS, and Perforce. Source control systems can easily merge, compare, and revert changes to VMware Project North Star (Thinstall) projects with no special steps. Another advantage of the .ini file format is that it is simple to read and modify from any programming language.

Compression

VMware Project North Star (Thinstall) is the only solution to provide block-based streaming decompression ("Block based Streaming" on page 53) directly into memory, which means compressed data does not need to be first decompressed to disk.

- You can launch brand new packages from a network share instantly without a lengthy decompression step. All the package data is decompressed a block at a time as needed by the application, so only startup data is sent over the network.
- Significant disk space savings are available on the client PC since file data is not decompressed to disk. When packages are deployed to PC hard drives for offline use, the disk requirements are much reduced because package data remains compressed at all time.
- VMware Project North Star (Thinstall) has similar compression ratios to ZIP

Beta

Streaming

VMware Project North Star (Thinstall) provides streaming capability without requiring a new server or a client. VMware Project North Star (Thinstall) uses the standard SMB protocol to stream applications over a LAN, so any Windows file share can instantly become a "streaming server". VMware Project North Star's embedded client technology means users can simply click on EXE files from network shares and the client will be loaded directly into memory.

- Client is Windows (already installed)
- Server is any SMB share (already exists)
- Streams block-by-block
- Packages over 8GB in size can start instantly
- Streams from any source media:
- Network shares & ISCSI
- Hard drive, USB Flash, CDROM

Terminal Server and MetaFrame support

- Typically the same VMware Project North Star (Thinstall) packages can be used on Terminal Servers and Desktop PCs without changes
- Ability to run multiple versions of application simultaneously on the same server without conflicts using registry and filesystem isolation
- Ability to update applications ("Upgrading applications" on page 57) without having to stop currently running copies
- Eliminates most problems with legacy applications relating to security permissions to the file system and registry

Sandboxing

- Application EXE remains "read only" at all times
- Stores application modifications on a per-user and per-application basis
- Prevents apps from "messing-up" machines
- Allows most non-Terminal Server apps to run on Terminal Server
- Allows most non-Vista apps to run on Vista
- Enables IT to maintain locked-down desktops
- "Reset" applications by deleting sandboxes

Virtual File system

- Only solution with no package size limits, supports packages over 4GB in size
- Supports sandboxing, isolation, and system merge modes for specific registry sub-trees
- Sandboxed files can be scanned/blocked by anti-virus software
- File system is stored in macro format, remapping shell folders on-the-fly ("Filesystem shell folder remapping" on page 20)
- File system automatically migrates SXS from XP to Vista

Virtual Registry

- EXE contains "read-only" image of registry
- Virtual registry is visible app only, not entire system
- Virtual registry is merged onto system registry.

Beta

- App specific registry sub trees are automatically isolated from the host PC to prevent conflicts with locally installed versions.
- Sandbox stores runtime modifications as “diffs”.
- Registry can contain per-user data (`HKEY_CURRENT_USER`)
- Registry can contain SID specific user data
- Registry can be reset to captured state by deleting sandbox
- Virtual registry is shared by all apps in the same sandbox
- Automatic backup and restore on disk corruption (common USB flash)
- Dynamic Path Relocation (“Dynamic Path Relocation” on page 19) applied to all registry read & writes allowing instant OS and PC migration

Virtual COM

- Support for “In process” COM from virtual DLLS.
- No system registration
- No extracting DLLs or OCXs to filesystem
- Support for “Out of process” COM/DCOM
- Use EXE based COM without install
- Support for “Services based” COM
- Use service based COM with virtual services

Dynamic Remapping

- Registry and file-system dynamically re-adjust for instant migration across OSs (“Dynamic Path Relocation” on page 19)
- User profile data and sandbox dynamically remaps on the fly allowing not just app migration but setting migrations
- Dynamic Remapping enables execution from USB flash from PC to PC
- Handles shell folders and short pathnames

Virtual Services (“Virtual Services” on page 24)

- Applications that require a service can be Thinstalled
- Virtual services auto-start when app is launched
- Virtual services can be started and stopped by application like real services
- Primarily for service-based license systems
- Support for packaging services and deploying as real Windows services started at boot time

Scripting

- Embed and execute .vbs in your EXEs
- Can pre-launch and shutdown
- Exposes VMware Project North Star (Thinstall) runtime API
- Access to any COM scripting provider
- Execute commands in virtual or real environment
- Can be used set application use time-outs, authenticate with LDAP or database

Beta

Active Directory Authentication

- Tie application access to AD Groups
- Dynamically add / remove users to AD groups to control access
- Prevent access from users not connected to domain
- Supports offline cached credentials for off-line usage

Packaging runtimes

VMware Project North Star (Thinstall) allows you to package any runtime you want to use directly with your application and eliminate pre-install requirements. Some examples include:

- .NET 1.1, 2.0, 3.0
- Java
- Perl
- Crystal Reports
- COM & ActiveX controls

Easy to deploy, stream, or push in EXE or .msi format

- Support for zero-footprint execution and streaming
- Easy desktop integration with filetypes and shortcuts using ThinReg
- Ability to generate self-registering .msi files ("MSI Generation" on page 93) enabling easy integration with desktop management products.

Packaging suites of applications

- Support for Reference EXEs
- Reference EXEs only take ~30k on disk
- Package multiple applications together for access to same shared environment
- Suites share the same virtual registry & file-system

Packaging ActiveX controls

- Ability to load Internet Explorer from host system using pre-fab virtual environment.
- Use ActiveX controls with out install
- Use Java-applets without installing Java

Runtime logging & diagnostics

- Report of potential errors in application
- Complete API trace with all parameters
- Top 100 slowest API calls
- DLL usage & load location
- Crash information
- COM Interface logging

Beta

User Mode

VMware Project North Star (Thinstall) is the only 100% user-mode application virtualization solution on the market. This document details why a user-mode solution is critical for maintaining system security, stability, and usability. One of the most fundamental principles of system administration is to use the Principle of least privilege (http://en.wikipedia.org/wiki/Principle_of_least_privilege), which requires the use of user-mode solutions where possible.

VMware Project North Star (Thinstall) can do the following things:

- VMware Project North Star (Thinstall) can run applications on locked-down PCs without Admin rights. This means users on the road can execute applications on kiosk and hotel PCs where they are not able to install software or device drivers.
- VMware Project North Star (Thinstall) can run application directly from USB Flash devices and various portable storage. Because VMware Project North Star (Thinstall) can be loaded in milliseconds without a client component, you can walk up to a PC and start running applications from a network share in seconds without messing around with any installation, support, and management of a client. VMware Project North Star (Thinstall) is the ultimate in application portability.
- VMware Project North Star (Thinstall) works well with almost all Windows environments from NT up to 64bit Vista.
- VMware Project North Star (Thinstall) plays well with all system-level software and doesn't conflict with various device drivers that may be added to your machines down the line.

Quick history of user-mode

Windows runs all code in one of two modes, user-mode and kernel-mode (also referred to as Ring3 and Ring0 respectively). The two modes reflect two different security models that are enforced directly by your Intel/AMD processor. Code running in kernel-mode has full machine access with no security controls, for example it can write to raw device ports, intercept and filter system-wide file system activity, read and write machine wide process and kernel memory, and access any kernel or process objects without regards to security descriptors. Kernel-mode code is either a device drivers or the windows kernel itself. User-mode is where all applications run and has strict security policies applied at all times, user-mode code cannot do anything to a machine that would directly cause it to crash or violate applied security policies. For example, user-mode code cannot access files owned by other users unless file system permissions allow it. As well user-mode code cannot make network connections unless the group security policy enables them to do so. 80386 (Pentium+) and higher processors enforce a specific security policy on user-mode application by prohibiting them from executing instructions that talk to device ports directly, preventing access to memory in other processes, and by preventing execution of kernel mode without first going through specific controlled entry points.

User-mode/kernel-mode are different from Administrator/Guest

For user-mode code, Windows supports many different sets of security policies based user accounts for example Administrator and Guest accounts are user-based security policies. Administrators typically have nearly full permission to access any system objects they want, while Guest are fairly restricted and cannot read other user's files or write to global locations on the computer. Both Administrators and Guest will run applications in user-mode, but switch to kernel-mode when making system calls to access objects. Kernel-mode code will then consult security descriptors to see if the user has access to the objects they request. Because Administrator accounts with full permission are a dangerous thing from a security perspective, in Windows Vista Microsoft reduced the rights of Administrators by default and will prompt users before accessing privileged objects on a computer.

Better security by confining vulnerabilities in user-mode

First, let's start with the assumption that all sufficiently large and complex software projects have some bugs and with many bugs there is the possibility for security exploit. The most public of these occur in Internet Explorer because it is easy to force every computer on the internet to execute code using Internet Explorer via HTML email or redirecting page views to illegitimate sites. With Windows Vista, Microsoft runs Internet Explorer in a separate user account which has limited security rights. This way, if IE is compromised, the compromised account cannot do anything to the rest of the machine. This solution is possible because IE runs

Beta

in user-mode. Because VMware Project North Star (Thinstall) runs 100% in user-mode, any bug or vulnerability presents no additional risk to the rest of the system because all VMware Project North Star (Thinstall) code is running in user-mode in the same security context as the application. Other solutions use device drivers and run significant amounts of kernel-mode code, which, if compromised allows full control over the machine. For a few examples, search for "symantec vulnerability" on google and you'll see this is a pretty common attack vector for hackers. User-mode code can be "walled-off" using user accounts, where-as kernel-mode code cannot. For this reason Microsoft recommends companies purchase user-mode solutions when possible, and put user-mode as a requirement in purchase RFPs:

http://technet.microsoft.com/en-us/windowsvista/aa906021.aspx

Also note MS reports user-mode solutions will help reduce your security risk:

"Running in standard user mode gives customers increased protection against inadvertent system-level damage caused by "shatter attacks" and malware, such as root kits, spyware, and undetectable viruses."

Better security by reducing attack surface

VMware Project North Star (Thinstall) is implemented 100% in user-mode and is only visible to a single application on the machine. Other solutions use device drivers which will intercept calls from every application on the system, if only to decide whether to filter the call or ignore it and pass on to windows. Because of this, the attack surface for VMware Project North Star (Thinstall) for security vulnerabilities is much smaller – vulnerabilities can only be exploited from the virtualized process and there is no possibility that non-virtualized Internet Explorer could represent a security threat because of low-level device drivers.

VMware Project North Star (Thinstall) keeps security policies intact

Because VMware Project North Star (Thinstall) runs 100% in user mode, it has the same rights and permissions as any other application a specific user would have. VMware Project North Star (Thinstall) has no ability to exceed the security rights of the user account it is running in because it has no device drivers or components running in kernel mode. When using VMware Project North Star (Thinstall), Administrators do not need to consider how their system-wide security policies will be affected. For example, other solutions used filter drivers to redirect file system activity, however it is not always clear what will be applied first; virtualization or security rights. VMware Project North Star (Thinstall) can be deployed in conjunction with central IT group policies or separately in the case of smaller development groups. In the later case, developer groups have the freedom to use the code components and frameworks of their choice without requiring security policy relaxation by centralized IT.

System stability through user-mode

Ever have your systems blue-screen, and you don't know why? They usually appear fairly random and can be hard to diagnose without significant time and resources. Microsoft reports that approximately 80% of system crashes and blue screens come from 3rd party device drivers. Every instruction running in kernel-mode represents an opportunity for a system crash or hang, often the hangs are due to thread dead-locks which only occur in rare circumstances and when combined with other products. It is a testing impossibility to ensure that products will never crash or hang, and most do. The difference is that user-mode code can never directly crash a system and it can be easily killed with the task manager if it hangs. There is no other impact on the system. Windows will automatically clean up after crashed or killed user-mode applications by closing open file handles, freeing allocated memory, and discarding created resources. Kernel-mode applications have no protection at all from crashing or leaked resources. A kernel-mode crash results in the dreaded blue-screen, though often times, the problem is a thread dead-lock and the machine appears to be frozen or performs very badly for no apparent reason. Diagnosing kernel-mode bugs is an art left to very expensive consultants and many companies find it cheaper to abandon faulty device drivers that crash systems rather than try to diagnose and resolve the problem.

Deployment to locked-down accounts without security elevation

Perhaps one of the biggest advantages of being a user-mode solution is that you can run applications directly from locked-down accounts without pre-installing a device driver or granting the user elevated security privileges. This has proved tremendously valuable for companies that have change management processes in place that prohibit the installation of device drivers or other system changes without a careful study, pilot program, and roll-back capability. Because VMware Project North Star (Thinstall) has no device drivers, it

Beta

enables deployment of applications with zero footprint on the desktop PCs, complex application can literally run directly from network shares with no preinstall requirements. This enables companies to guarantee change management policies are not violated through group policies, and quickly deploy new applications at the same time. Because there are no device drivers or system files installed, the roll-back process is as simple as deleting an EXE file.

Ok, user-mode is great—why is VMware Project North Star (Thinstall) the only one doing it?

Thinstall pioneered the user-mode application virtualization with patents granted and pending. The user-mode virtualization technology is very large undertaking to complete. VMware Project North Star (Thinstall) has been in development for nearly 10 years to date, and it is technologically most similar to the open source projects Wine and ReactOS which have also been in development for approximately the same time frames. Thinstall focused on staying user-mode from day one, and this is reflected in the product you see today. Not knowing a user-mode solution was possible, other solutions started out as device driver based solutions and switching to a user-mode model would require complete rewrites. Device driver solutions work by implementing a windows filter driver in kernel mode which intercepts various file-system requests in the kernel mode. At this level, the hard work of loading applications, resolving DLLs, and processing side-by-side manifest is done by the Windows operating system. Virtualization at the filter-driver level is comparatively simple to implement (the Vista OS team has implemented this in the Windows Vista kernel). In user-mode, virtualization is much more complex and requires virtualization of any functionality that touches the file system or registry. More than 400 different windows sub-systems and APIs have been replaced by VMware Project North Star (Thinstall) including the windows loader, service control manager, and process manager.

Side-by-Side (SxS)

SxS is a relatively new OS feature supported by Windows on XP, Windows 2003, and Vista. SxS is a must-have feature in order to deploy most modern software products such as Microsoft Office 2007, Adobe Reader 8, and .NET 2.0/3.0. VMware Project North Star (Thinstall) is the only application virtualization solution which supports SxS.

In the early days Microsoft recommend installing DLLs to `c:\windows\system32` if you wanted to share them with other applications. This lead to "DLL Hell" since one applications installer would install one version of a DLL and another installer would install another version overwriting the existing DLL. If the new version of the DLL wasn't compatible the older application would suddenly break and stop working.

This is still how the majority of applications install their DLLs, even Office 2007 installs approximately 7 DLLs to `c:\windows\system32`. Slowly the industry is moving to deploy shared DLLs using SxS as XP becomes mainstream and Visual Studio 2005 forces this on you if you want to use the Version 8 C Runtime libraries.

With SxS technology, applications can install DLLs to version specific directories tell Windows what version of the DLL should be used when they load a DLL by that name. For example, Office 2007 installs MFC80 (8.0.50727.42) to the following path:

```
%SystemRoot%\WinSxS\x86_Microsoft.VC80.MFC_1fc8b3b9a1e18e3b_8.0.50727.42_x-ww_dec6ddd2\mfc80.dll
```

If you wonder about the weird path, it contains information about the DLL version. The algorithm for producing this path is purposefully unpublished by Microsoft so the only way to place files here is by using the MSI installer technology. Because MSI is responsible for putting SxS Dlls in the correctly location, Microsoft was free to tinker with the path format in Windows Vista. So a installed SxS dll will go to different locations on Windows XP and Windows Vista. Because users might migrate their existing XP installations to Vista, the upgrade process will automatically move the DLLs to their new home on Vista.

Windows OSes which have SxS will look for an application's manifest information to determine which version to load from SxS. A manifest is a fairly simple XML description of all the SxS DLLs that might be loaded by the application and which version to use. The manifest can either be embedded in the application EXE/DLL file as a resource or stored separately on the filesystem as a `.manifest` file. In the second case, the manifest file for `app.exe` would be called `app.exe.manifest`.

Beta

The manifest resolution algorithm is fairly complex and I won't get into all the details here, but it can include among other things searching based on the user's locale (i.e. Look for French/German version first). As well, to eliminate security disasters like what happened with `gdiplus.dll`, SxS DLLs have a "force upgrade" mechanism implemented as separate security policy files (also XML files). This mechanism allows Microsoft to override version number request by applications in the case where some gaping security hole is discovered in a shared library that effects thousands of applications. For example, an application manifest may say "I want version 1.0.0.0 of xyz.dll", but Microsoft can push a policy file update that says "redirect all request of version 1.0.0.0 to 1.0.0.5".

VMware Project North Star (Thinstall) has built-in support for SxS and manifest/policy file handling, so if you capture the installation of an application that installs SxS dlls it works perfectly on every platform with zero installation.

If the application normally installs DLLs to `c:\windows\winsxs\...` this will be reflected in the capture and show up in your project under `%systemroot%\winsxs\....`

At runtime, VMware Project North Star (Thinstall) will examine the application's manifest file (from the resource section or separate `.manifest` virtual file) and determine which version of a DLL should be used. VMware Project North Star (Thinstall) then loads the correct version of the DLL from the virtual path (contained inside the package). VMware Project North Star (Thinstall) supports management of runtime activation contexts as well, so it knows which version of DLLs to load from dynamic dll loads.

Natively (without VMware Project North Star) SXS is only supported on xp, w2k3, and vista, but not 2k and NT. This means if you want to support 2k and NT (without VMware Project North Star) you need to install into 2 locations (automatically done by MSI installer 3.0):

1 `c:\winnt\system32` (or application directory)

2 SXS path location (`c:\winnt\winsxs`)

Location #1 is required to support nt/2k, and location #2 is required in order to meet MS's "Designed for Windows" guidelines which state the app must continue to work if the OS is upgraded from 2k->xp

Because VMware Project North Star (Thinstall) has its own SxS processing, all SxS technology is available for both nt/2k and xp/w2k3/vista in the Thinstalled environment. That's right, we bring modern OS technology and make it work on older platforms!

On Windows Vista, Microsoft has changed the path locations and file path name-mangling algorithm for SxS DLLs. VMware Project North Star (Thinstall) automatically moves SxS DLLs captured for XP/w2k3 and moves them to the correct location in virtual space for Windows Vista at runtime when the app starts – so the same EXE can run on all platforms with no changes. This is why we advertise VMware Project North Star (Thinstall) as "instant migration".

In the case of emergency policy updates, VMware Project North Star (Thinstall) will automatically switch from using the packaged version of DLL to system DLLs if the system policy file dictates. This means if the host PC is up to date with Windows update, you are protected from Thinstalled packages that contain DLLs that fall into this "needs emergency update" category. When we implemented this, we had a discussion with the Microsoft SxS team, and they assured us that policy files are only used in 2 cases:

1 In case of emergency (none have occurred yet)

2 For MS beta products, in this case beta DLLs will be redirected to shipped versions

VMware Project North Star (Thinstall) is the only virtualization technologies which fully supports SxS which allows you to virtualize most new applications where virtualization products fail and must have SxS dlls physically installed on the machine before they will work.

Dynamic Path Relocation

An important feature and concept of VMware Project North Star (Thinstall) is *Dynamic Path Relocation*, which refers to the ability to move files and modify registry values to match the local host PC on the fly.

Beta

VMware Project North Star (Thinstall) performs dynamic remapping during application startup and during runtime which allows both applications and their associated settings to migrate across different versions of Windows.

Instant Side-by-Side DLL migration

Windows XP Side-by-Side DLLs are migrated to Windows Vista Side-by-Side DLLs automatically depending on the platform you are running on

If you capture an application that uses SxS DLLs on NT/2k/XP/w2k3 it will install SxS DLLs to a path location differently than when installed on Vista. VMware Project North Star (Thinstall) dynamically moves these SxS during the application startup if the platform has changed. Using VMware Project North Star (Thinstall)'s Dynamic Path relocation, you can create one package that works on all platforms.

Filesystem shell folder remapping

Many applications will access files using Shell folder locations, for example applications typically call GetWindowsDirectory to obtain the path to `c:\windows` instead of using a hard coded path. On different versions of Windows the system directory will be located in different locations. For example `c:\winnt` is used for NT and Windows 2000 by default, as well the user can select an alternate directory during installation of Windows. Applications will also typically use `shfolder.dll` to obtain the path various "shell folder" locations like `c:\program` files and `c:\documents` and `settings\username`. For a complete list of shell folder paths, see "Folder Macros" on page 148.

An example is Macromedia flash which installs to `c:\windows\system32\macromed\flash`. At runtime, flash uses GetWindowsDirectory to obtain the partial path `c:\windows\system32`, and then appends `macromed\flash` to obtain the location of it's install directory. As well, flash uses a registry value which corresponds to this location:

```
HKEY_CLASSES_ROOT\CLSID\{1171A62F-05D2-11D1-83FC-00A0C9089C5A}\InprocServer32 DefaultValue =
C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx
```

When running an application that installs Macormedia Flash dynamically or during the capture process (for example Firefox or Internet Explorer), the registry stores the path `C:\WINDOWS\system32` and files are written to `c:\windows\system32\macromed\flash`. If the application is moved to another PC where the windows root directory is different (for example `c:\winnt windows 2000`), it will fail to work unless both the files and registry keys are remapped to point to `c:\winnt`.

VMware Project North Star's virtual filesystem stores file paths using Folder Macros ("Folder Macros" on page 148), so the file paths will automatically expand to the correct location on different PCs. As well, VMware Project North Star (Thinstall) stores registry data using the same Folder Macros, so that registry values automatically re-adjust to point to the correct location on a different PC. For example, when the application writes the registry value `C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx`, VMware Project North Star (Thinstall) stores this internally as `%SystemSystem%\Macromed\Flash\Flash9b.ocx`. When the application queries for this value, it will transparent expand back to `C:\WINDOWS\system32\Macromed\Flash\Flash9b.ocx` when running on Windows XP/2k3/Vista and `C:\winnt\system32\Macromed\Flash\Flash9b.ocx` when running on Windows 2k and NT.

Short pathname remapping

VMware Project North Star (Thinstall) has unique support for handling short pathnames which allows applications to be installed to their default location and migrate across PCs. For detailed information see "Short Pathnames" on page 21.

Registry data remapping

VMware Project North Star (Thinstall) intercepts all data written to the registry and sees if there are references to short pathnames or shell folders. If there are, it will internally store the registry data in macro format so that the data re-expands to the correct location on other PCs. This functionality is described in more detail in the Short Pathnames sections ("Short Pathnames" on page 21).

Beta

Short Pathnames

Short pathnames are DOS 8.3 compatible names that map into their long pathname equivalents. For example `C:\PROGR~1` is the short pathname version of `C:\Program` Files.

Short pathnames are important for most applications for several reasons:

- Short pathnames eliminate spaces

  Short pathnames eliminate spaces from paths which prevents some compatibility and security issues. For example, when executing the command `C:\Program Files\Microsoft Office\OFFICE11\winword.exe c:\Myfile.doc` using ShellExecute, Windows will actually try a number of possible commands to see if they are valid:

  ```
  c:\Program.exe "Files\Microsoft Office\OFFICE11\winword.exe c:\Myfile.doc"
  c:\Program Files\Microsoft.exe "Office\OFFICE11\winword.exe c:\Myfile.doc"
  c:\Program Files\Microsoft.exe Office\OFFICE11\winword.exe "c:\Myfile.doc"
  ```

  If a user creates the file `c:\Program.exe` on a PC, it could break office and also represents a security threat.

- Short pathnames allow compatibility with legacy applications

  16 bit applications cannot support filename paths other than 8.3 style. When windows runs 16bit applications, it will provide applications with the short pathname version for all filenames. This typically does not come into play for most people, but was one of the primary reasons for the original conception of short pathnames.

- Short pathnames work-around Windows API path length limitations

  Many Windows API functions have limitations on the maximum string length they can handle for paths. For example on Windows XP SP1, the ShellExecute API command cannot handle a string length longer than 128 characters. Often the registry is used to store a value that look like "Command + Parameters" like this entry for Office 2003:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Applications\ois.exe\shell\Edit\command DefaultValue =
  C:\PROGRA~1\MICROS~3\OFFICE11\OIS.EXE /shellEdit "%1"
  ```

  If the Exe filename plus the parameter is greater than 128 characters, the command will fail on XP SP1 (Microsoft extended the length for ShellExecute in SP2). Considering the user is likely to use a filename from their "My Documents" folder, and the filename may be significantly long—if you don't use short pathnames there is a very high chance of failure.

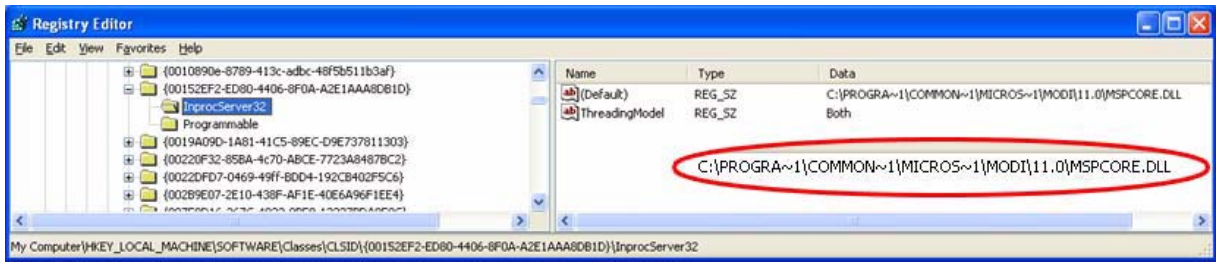
  For example, this command exceeds 128 characters (168 chars):

  ```
  C:\Program Files\Microsoft Office\OFFICE11\winword.exe c:\documents and settings\Greogory
  Appleblaught\My Documents\Draft 34 April 6 2006 Master License Agreement.doc
  ```

  The same path is much smaller using short pathnames (85 characters):

  ```
  C:\Progra~1\Micros~3\OFFICE11\winword.exe c:\docume~1\Greogo~1\MyDocu~1\Draft3~1.doc
  ```

The use of Short pathnames in the registry is very common, for example Office installs hundreds of short pathname references in the registry. Below you can see a few examples from an Office 2003 install.



Out of process COM

```
HKEY_CLASSES_ROOT\CLSID\{00020800-0000-0000-C000-000000000046}\LocalServer32
              C:\PROGRA~1\MICROS~3\OFFICE11\GRAPH.EXE /automation
```

Beta

In process COM

```
HKEY_CLASSES_ROOT\CLSID\{0002E55E-0000-0000-C000-000000000046}
               C:\PROGRA~1\COMMON~1\MICROS~1\WEBCOM~1\11\OWC11.DLL
```

Cryptography

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Exchange
Cryptographic Provider v1.0
                C:\PROGRA~1\MICROS~3\OFFICE11\EXCHCSP.DLL
```

Intelligent Search

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Intelligent Search\Schema\1.0\XML\MSUSP File1 =
               C:\PROGRA~1\MICROS~3\OFFICE11\USPTYPES.XML
```

Internet Explorer plugins

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet
Explorer\Extensions\{92780B25-18CC-41C8-B9BE-3C9C571A8263}
                Icon = C:\PROGRA~1\MICROS~3\OFFICE11\REFBAR.ICO
```

Location of shared DLLs

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\11.0\Common\FilesPaths
               mso.dll = C:\PROGRA~1\COMMON~1\MICROS~1\OFFICE11\MSO.DLL
```

Error reporting and crash handling

```
HKEY_LOCAL_MACHINE\Software\Microsoft\PCHealth\ErrorReporting\DW\Installed
               DW0200 = C:\PROGRA~1\COMMON~1\MICROS~1\DW\DW20.EXE
```

Short pathnames are not consistent across PCs and are also highly dependent on installation order. For example, installing Microsoft Office after installing Microsoft Visual Studio can create identical short pathnames which refer to different location than when installing Microsoft Visual Studio and the Microsoft Office:

installing Microsoft Office and then Microsoft Visual Studio generates:

```
c:\progra~1\micros~1 and c:\progra~1\micros~2

c:\progra~1\micros~1 = c:\program files\Microsoft Office
c:\progra~1\micros~2 = c:\program files\Microsoft Visual Studio
```

installing Microsoft Visual Studio and then Microsoft Office also generates the same two paths which refer to different locations:

```
c:\progra~1\micros~1 and c:\progra~1\micros~2

c:\progra~1\micros~1 = c:\program files\Microsoft Office
c:\progra~1\micros~2 = c:\program files\Microsoft Visual Studio
```

When capturing a snapshot of an application's install, it will write a number of short pathname values to the registry as described above. When moving the application to a different PC or installing additional applications on the same PC may effect the short pathname values that the operating system provides from the underlying operating system. Virtualization solutions that are based on filter drivers do not have ability to control short pathname values because they are generated by the windows filesystem and as a result a capture on one PC has a good chance of failing when moved to another PC or when executed some time later on the same PC when other applications have been installed. Common areas of failure that occur when proper short pathname support is not available include:

- In-process and out-of-process COM failure.

  The application tries to create COM objects and fails because the registry values point to the in-process or out-of-process COM server no longer point to the correct location.

- MSI is executed to reinstall the application even though it was fully installed.

Beta

`msi.dll` performs an integrity check for all msi installed components requested by the application. If `msi.dll` notices a DLL or data file is not located at the same location as it was originally installed to, it will kick off a reinstall procedure to try to correct the problem.

- Failure to execute child processes

  Many child process applications are executed using registry values that point to short pathnames. If the values do not point to the correct location, the child process will fail to launch. This will often render the application unusable.

- Application fails to load

  Applications may crash or fail to load because the cannot locate there installation path or shared DLL paths using registry data.

- Application complains of missing data files

  The application may display error messages relating to missing files because it can no longer locate a file using a short pathname pointer from the registry.

How VMware Project North Star (Thinstall) solves short pathname Migration

VMware Project North Star (Thinstall) uses a combination of dynamic registry data expansion and virtual short pathnames to enable applications to instantly migrate from one PC to another. When an application writes a value to the registry, VMware Project North Star (Thinstall) will scan the data to see if it references a short pathname or shell folder location. If it does, VMware Project North Star (Thinstall) will store it internally in macro format which encodes both the shell folder location and the long pathname value for a short pathname. At runtime, when the application queries the same registry value it will receive the original value it wrote to the registry. If the application moves to a different PC where the short pathnames are different it will obtain an automatically adjusted value.

For example, if you look at `HKEY_LOCAL_MACHINE.txt` for a capture of Microsoft Office you'll see some entries that look like this:

```
isolation_full HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{03B54468-0899-4233-
               8689-623FFFC295EE}\InprocServer32
               Value= REG_SZ~%ProgramFilesDir~0032\Common Files\Microsoft Shared\Smart
               Tag\IETAG.DLL#2300
               Value=ThreadingModel
               REG_SZ~Apartment#2300
```

In this case the registry value is encoded to expand to the shell folder location `c:\program files` plus `\common files\microsoft shared\smart tag\ietag.dll` like the following expression:

```
GetShortPathName(ExpandMacro("%ProgramFilesDir%") + "\Common Files\microsoft shared\smart
tag\ietag.dll")
```

Because VMware Project North Star (Thinstall) performs this macro expansion and collapse for every registry operation, it has been highly optimized to require very little CPU overhead. The expansion and collapse occurs dynamically at runtime so there is no startup overhead time for applications with large registries.

Preventing conflicts when Isolation is used

In a virtual environment where the application is isolated from the PC, the normal file system will not be aware of short pathnames that a virtualized application wants to use. For example consider the following scenario:

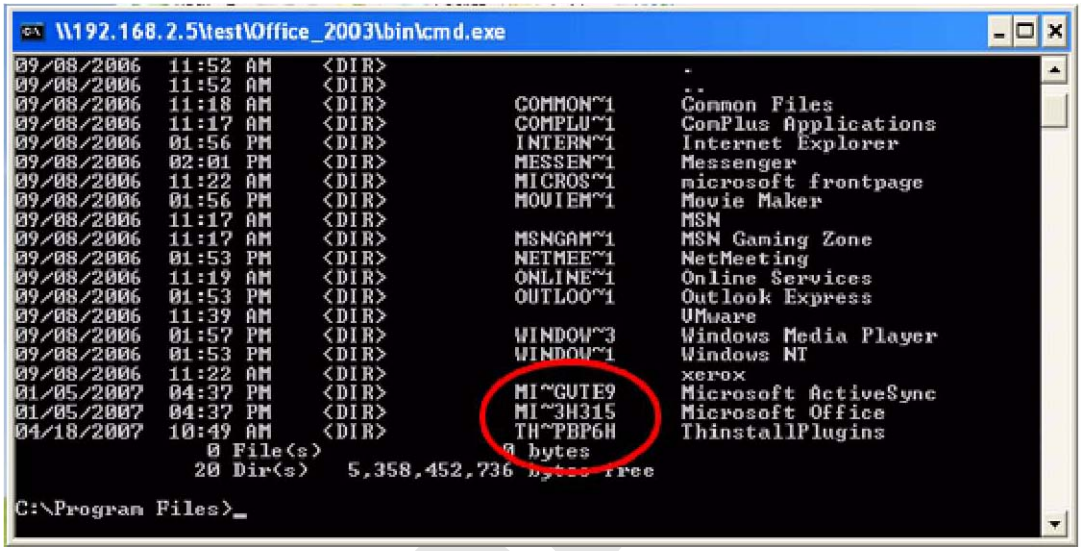
A desktop PC has Microsoft Office Installed installed natively (no virtualization)

In this case Microsoft Office occupies the short pathname `c:\progra~1\micros~1`

On this same desktop, a virtual version of Microsoft Visual Studio application begins running... In this case Visual Studio needs to use a different short pathname other than `c:\progra~1\micros~1` since this is in use. The application must reserve the short pathname `c:\progra~1\micros~2` when short pathname is allocated by the Windows filesystem. A problem arises if a 3rd "Microsoft" application is installed while Visual Studio is running, will it get microso~3 or microso~2? In the later case, it will conflict with the virtual application and the virtual application may fail.

Beta

VMware Project North Star (Thinstall) does not use filter drivers and does not depend on the Windows file system for the allocation of short pathnames. VMware Project North Star (Thinstall) prevents conflicts between short pathnames between virtual applications and system applications by using it's own short namespace which will not collide with system applications. Since each virtual application is isolated from all other virtual applications there is no problem with having 2 virtual application using the same short pathnames internally. In the screen shot below you can see VMware Project North Star (Thinstall) uses a different name space than Windows for short pathnames when the system does not have the paths already created. You can see this in action by running a virtualized `cmd.exe` and use `dir /x` to list short pathnames. The three directories below do not follow the normal Windows convention of ABCDEF~1.


\\192.168.2.5\test\Office_2003\bin\cmd.exe
09/08/2006 11:52 AM <DIR> .
09/08/2006 11:52 AM <DIR> ..
09/08/2006 11:18 AM <DIR> COMMON~1 Common Files
09/08/2006 11:17 AM <DIR> COMPLU~1 ComPlus Applications
09/08/2006 01:56 PM <DIR> INTERN~1 Internet Explorer
09/08/2006 02:01 PM <DIR> MESSEN~1 Messenger
09/08/2006 11:22 AM <DIR> MICROS~1 microsoft frontpage
09/08/2006 01:56 PM <DIR> MOVIEM~1 Movie Maker
09/08/2006 11:17 AM <DIR> MSN
09/08/2006 11:17 AM <DIR> MSNGAM~1 MSN Gaming Zone
09/08/2006 01:53 PM <DIR> NETMEE~1 NetMeeting
09/08/2006 11:19 AM <DIR> ONLINE~1 Online Services
09/08/2006 01:53 PM <DIR> OUTLOO~1 Outlook Express
09/08/2006 11:39 AM <DIR> VMware
09/08/2006 01:57 PM <DIR> WINDOW~3 Windows Media Player
09/08/2006 01:53 PM <DIR> WINDOW~1 Windows NT
09/08/2006 11:22 AM <DIR> xerox
01/05/2007 04:37 PM <DIR> MI~GUIE9 Microsoft ActiveSync
01/05/2007 04:37 PM <DIR> MI~3H315 Microsoft Office
04/18/2007 10:49 AM <DIR> TH~PBP6H ThinstallPlugins
0 File(s) 0 bytes
20 Dir(s) 5,358,452,736 bytes free
C:\Program Files>_

Result: Install to default location, migrate seamlessly

Other virtualization solutions require you to install applications to separate drive letters using 8.3 filenames in order to resolve the various short pathname issues described above. Because many applications will not let you specify an alternate installation path or will not run correctly when installed to a non-default location, this approach can only be used in limited situations. When using VMware Project North Star's Setup Capture ("Setup Capture" on page 69) to create Thinstalled applications, you can install applications using their default location without concern for short pathnames.

Virtual Services

VMware Project North Star (Thinstall) can be used to package up services in 2 different modes:

1 A Virtual Service

- Packaged together with a main supporting application
- Runs on application start-up
- Runs under the user account which ran the application
- Can be started and stopped only by virtual applications in same sandbox (Visible to services control panel services manager only by running from virtual `cmd.exe`
- Zero Installation or system changes required

By default any application that installs a service will automatically use virtual services. When using Setup Capture to capture an application which installs virtual services, *no changes are needed to your project to run virtual services.*

Starting virtual services

VMware Project North Star (Thinstall) uses the registry values captured by Setup Capture to determine if it should automatically start a virtual service when it's host application is run.

If the *Service Startup Type = Automatic*, then VMware Project North Star (Thinstall) will start the service automatically before executing the host application.

If the *Service Startup Type = Manual*, then VMware Project North Star (Thinstall) will not start the service until a host application specifically starts the service using service API calls (OpenService/StartService). If you want to prevent virtual service from being automatically started when the first parent process is launched, you can use the AutoStartServices option ("AutoShutdownServices" on page 98).

Stopping virtual services

Virtual Services that are started during application startup will be automatically shutdown when the last non-service application exits. You can force virtual service to continue running until the user logs off using the AutoShutdownServices option ("AutoShutdownServices" on page 98).

running until the currently logged in user logs out unless the application explicitly tells the service to stop. See "Stopping Service" on page 119 script for an example of how to explicitly stop virtual services when the main application exits.

NOTE This behavior may change in future version of VMware Project North Star (Thinstall).

---

2 A real "Real" Windows Service

- Packaged as a separate Thinstalled EXE file which can be run directly
- Runs on boot-up
- Runs under the account specified by the service
- Will be directly visible to the control panel services manager
- Can be started and stopped by any application
- Requires global system registry change

Running Thinstalled EXE as "Real" Windows Services

For some applications, it may be desirable to run as a real windows service on system boot up rather than when an application is launched by a user.

To install a Thinstalled packaged EXE as a real service, the general procedure looks like this

1 Remove virtual registry keys from your VMware Project North Star (Thinstall) project relating to services

2 Add real system registry keys for the associated service which point to your VMware Project North Star (Thinstall) packaged EXE

More specific instructions using Apache web server as an example:

1 After capture delete all services registry keys from `HKEY_LOCAL_MACHINE.txt`
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`

2 On target machine where you want to run the service, write real registry keys where ImagePath points to your Thinstalled EXE, like this:

```
c:\path_to_thinstalled_apache\httpd.exe –k runservice
```

You can write the registry values using a `.reg` file or as part of an installation process.

3 You may need to reboot before the service will show up in the service control panel (**Start>Run>services.msc**). Once it shows up, you can test starting and stopping the service directly by using the control panel.

NOTE rebooting isn't required to use the service, it's just to have the service control panel refresh it's list of available services.

Below are the registry keys one would delete from `HKEY_LOCAL_MACHINE.txt` for the apache web server.

Beta

```
isolation_full HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Apache2.2
                Value=Type
                REG_DWORD=#10#00#00#00
                Value=Start
                REG_DWORD=#02#00#00#00
                Value=ErrorControl
                REG_DWORD=#01#00#00#00
                Value=ImagePath
                REG_EXPAND_SZ~"%ProgramFilesDir%\Apache Software
                Foundation\Apache2.2\bin\httpd.exe" –k runservice#2300
                Value=DisplayName
                REG_SZ~Apache2.2#2300
                Value=DependOnService
                REG_MULTI_SZ~Tcpip#2300Afd#2300#2300
                Value=DependOnGroup REG_MULTI_SZ=#00
                Value=ObjectName
                REG_SZ~LocalSystem#2300
                Value=Description
                REG_SZ~Apache/2.2.3 (Win32)#2300
isolation_full
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Apache2.2\Parameters
Value=ConfigArgs REG_MULTI_SZ~–f#2300%ProgramFilesDir%\Apache Software
Foundation\Apache2.2\conf\httpd.conf#2300–d#2300%ProgramFilesDir%\Apache Software
Foundation\Apache2.2\.#2300#2300

isolation_full HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Apache2.2\Security
Value=Security
                REG_BINARY=#01#00#14#80#90#00#00#00#9c#00#00#00#14#00#00#00#30#00#00#00#02#00#
                1c#00#01#00#00#00#02#80#14#00#ff#01#0f#00#01#01#00#00#00#00#00#01#00#00#00#00#02#
                00#60#00#04#00#00#00#00#00#14#00#fd#01#02#00#01#01#00#00#00#00#05#12#00#00#00
                #00#00#18#00#ff#01#0f#00#01#02#00#00#00#00#00#05#20#00#00#00#20#02#00#00#00#00#14
                #00#8d#01#02#00#01#01#00#00#00#00#00#05#0b#00#00#00#00#00#18#00#fd#01#02#00#01#0
                2#00#00#00#00#00#05#20#00#00#00#23#02#00#00#01#01#00#00#00#00#00#05#12#00#00#00#
                01#01#00#00#00#00#00#05#12#00#00#00

isolation_full HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Apache2.2\Enum
                Value=0
                REG_SZ~Root\LEGACY_APACHE2.2\0000#2300
                Value=Count
                REG_DWORD=#01#00#00#00
                Value=NextInstance
                REG_DWORD=#01#00#00#00
```

Change Log

Version 3.330, Dec 14 2007

- Fixed a problem where error messages were not displayed correctly in SetupCapture
- Save drives to be scanned by SetpuCapture across reboots
- Fixed generation of incorrect VirtualDrives lines in `Package.ini` (fixes licensing of QuickBooks 2008)
- Fixed problems when using `D:\Program Files` instead of `C:\Program Files`
- Added ExternalDLLs option ("ExternalDLLs" on page 103)
- Respect suppression of error messages requested by app when loading a DLL fails
- Fixed problem which caused long delays in Vista-style Open File dialogs when browsing to network shares
- Fixed MUI problem with DialogBoxParam
- Allow surrounding spaces in section and key names for .ini files
- Fix IE6
- SxS now looks up the source directory for dependants

- Fixed problem with SQL Server Configuration Manager
- Fixed problem with clip art not showing up in Word 2007 on Vista
- Fixed problem with opening Word and Excel attachments in Outlook 2007 where a .vbs script was included
- Fixed a problem with automatic update of Visio 2007
- Fixed a number of problems related to Merged isolation mode of the registry
- Fixed problem with connecting to remote registry
- Fixed problem with icon extraction functions
- Properly recognize MS-style short names with an extension
- Timeout sooner on service-implemented COM object creation
- Exclude Thinstall directory from SetupCapture so we won't be including snapshot files in the package
- Fixed problem with SxS manifests on Vista
- Fixed problem where Crystal Reports would display an error message "Can't find module"
- Improve support for editing of OLE embedded objects
- Fixed problem with memory mapped files which caused Autodesk Revit 2008 to crash during startup
- Have thinreg remove intermediate directories on uninstall
- Fixed problem where captured files on extra drives may not be visible on other machines
- Fixed problem where permission is denied during registry snapshot
- Prevent unnecessary copying of .exe files to the sandbox by ShellExecute
- Make tlink search its own directory for thinreg.exe when generating .msi files
- Add support for REG_QWORD in the build tools
- Enable window theming by apps which use SetWindowsHookex for this
- Added documentation for BlockSize ("BlockSize" on page 99) and ExcludePattern ("ExcludePattern" on page 102)

Version 3.300, Oct 30 2007

- Move BuildCache to local appdata so it won't be copied on login/logout for roaming profiles
- Fix launching of external document viewers from Outlook
- Added SetFileSystemIsolation ("SetFileSystemIsolation" on page 127) and SetRegistryIsolation ("SetRegistryIsolation" on page 128) API functions
- Added ReserveExtraAddressSpace option ("ReserveExtraAddress..." on page 110)
- During virtual app startup, register fonts installed by the app even if they were installed outside the `%Fonts%` directory
- Fix `.ini` file handling on MBCS systems
- Improve handling of MS-style short pathnames
- Pass service manager calls for other hosts on to Windows
- Handle MUI files on Vista
- Fix issue where services shutdown would get stuck in an infinite loop
- Fix issue where DLLs loaded with `LOAD_WITH_ALTERED_SEARCHPATH_FLAG` were not found

Beta

Version 3.210,Sept 11 2007

- Added MSI Generation capability (Chapter 7, "MSI Generation," on page 93)
- ThinReg updated to support unregistration via command-line
- ThinReg updated to support specification of filetypes not captured in original package
- ThinReg fixed to support some file types associations that did not previously work

Version 3.203,Sept 11 2007

- Updated support for newer cygwin tools which use NtCreateFile/NtReadFile directly
- Added ThinReg utility ("ThinReg" on page 86) for automatic Shortcut & File type associations

Version 3.146, Aug 27 2007

- Automatic configuration of Office using Office Customization Tool
- Added script function AddForcedVirtualLoadPath which indicates VMware Project North Star (Thinstall) should load all DLLs in a directory instead of Windows even if the DLLs exist outside of the Package
- Fixed tlink, vftool, and vregtool so they don't pause when not run from a console
- Add `Package.ini` setting to override version resource strings ("Version.XXXX" on page 115)
- Don't show EULA for console processes running without visible console
- Fix creation of Exchange accounts in Outlook
- Added isolation options for Shared memory objects ("IsolatedMemoryObjects" on page 104) and synchronization objects ("IsolatedSynchronizatio..." on page 105)
- Added ability to control sandbox path in `Package.ini` ("SandboxPath" on page 112)
- Added more documentation for `Package.ini` options (Chapter 8, "Package.ini format," on page 95)
- Fixed issue where virtualized IE6 may lockup because of shared memory object with `explorer.exe`
- Fixed issue where `comctl32.dll` was not able to load AVI files from virtual filesystem
- Fixed issue where IE6 crashes on startup on Vista (still has a few other issues on Vista)
- Fixed problem where SearchPath fails for paths that absolute paths that start with "\"
- Fixed issue where options set in [BuildOptions] are not overwritten by options set in individual [`App.exe`] sections
- Added RegConnectRegistry
- Fixed problem where SxS would load the incorrect version of system DLLs when policy redirects are in place
- Fixed issue where Outlook 2007 is unable to change signatures or stationary/fonts
- Partial support for Debug API
- Fixed issue with Petrel.1 where DLL sections have both "uninitialized" and "initialized" flags set
- Fixed issue where single-use out-of-process COM servers were not being unregistered after use
- Fixed issue where apps with DEP enabled would be terminated by Vista when exceptions occur
- Added VirtualDrives support ("VirtualDrives" on page 114), this is automatically added to projects by SetupCapture

Version 3.128, July 11 2007

- Fix account creation wizard in Outlook. Also fixes some MSI installer popups on first run of Office apps
- Fix hang of Adobe Reader 8.1 on machines which don't have .NET installed

Beta

- Auto-start virtual services even when those services are installed on the physical system too

Version 3.127, July 9 2007

- Improved GUI for selecting user-accessible EXEs to build
- Setup Capture will pickup application command-line parameters and alternate icon automatically
- Fixed outlook lock-up issue
- Fixed issue with cygwin stack
- Added AutoShutdownServices BuildOption for `Package.ini`.

NOTE the default behavior is now to shutdown services when the last non-service process exits, the old behavior was to keep services running. If you want the old behavior, add "AutoShutdownServices=0" to the [BuildOptions] section in your `Package.ini` file

---

- Fixed issue where Outlook 2003 would show an error "Can't find Outlook:today" on Windows 2003
- Fixed hang of Adobe Reader 5 when running as Guest on Windows 2000
- Don't hardcode `C:\Program Files` in the Setup Capture-generated `build.bat`, use the correct localized directory

Version 3.118, June 23 2007

- Fixed issue where .net 2.0 ngen images would crash when relocated in memory
- Fixed installer to support NT4
- Fixed registry handle leak
- Support command-line arguments for 16bit apps
- Fixed problem where some directories were not visible to 16bit apps as 8.3 paths
- Some fixes for Outlook 2007 on vista
- Fixed issue where fonts registered during install may not be visible
- Resolves unicode issue with registry values and subkeys
- Improved support for msi installer service

Version 3.111, June 6 2007

- Fixed problem where deleting system file using a shortname will cause a subsequent create operation to fail
- Fixed issue where "Could net verify license" message box appears for some child processes
- Fix for Autodesk DWG Trueview 2007

Version 3.104, May 23 2007

- Added Sandbox Merge Utility ("sbmerge" on page 85), allows you to merge sandbox modification back into a project
- Fixed issue with Photoshop CS3, may fix some other adobe products
- Experimental support for Win3.1 (16-bit) applications (full support with 3.2)
- Fixed out-of-process COM issues
- MSI Installer service is disabled for this release, we'll have better support in the next release
- Fixed problem where logging would lock-up on Vista
- Fixed Thread local storage problem
- Fixed issues with SQL Express 2005

Beta

Version 3.085, May 3 2007

- Fixed issues with MSDE
- Fixed problem with launching Outlook control panel
- Fixed problem where there is a 20 second pause when services do not startup
- Fixed problem where .ini files may be written to if they pre-exist in on the system
- Fixed problem where load/unload/load sequences for some DLLs could cause a crash
- Added support for RegGetValue (Vista)
- Fixed issue where Thread-local storage was not being allocated for threads created for services
- Added support for SetServiceBits
- Fixed issue where applications call registry functions with invalid buffer sizes

Version 3.081, Apr 9 2007

- Fixed short path problem where MSI thinks products are not installed correctly when it obtains two different shortpath values
- Fixed problem where applications calling registry queries with a buffer larger than 1GB of memory would cause a lock-up

Version 3.080, Apr 9 2007

- Now you can easily tell which version of VMware Project North Star (Thinstall) an EXE was built with by looking in the File Version information pane
- Added `Package.ini` option (`RemoveSandboxOnExit=1`) to delete sandbox after execution completes (Chapter 8, "Package.ini format," on page 95)
- Added scripting callback function (`OnLastProcessExit`) which will be called when the last process quits Chapter 9, "Scripting," on page 117
- Added ability to perform in-places updates to application while EXE files may be locked by users ("Upgrading applications" on page 57)

Version 3.078, Apr 5 2007

- Update to correct Office XP DEP related bug
- Fixed problem with 3.077 where licenses could not be verified

Version 3.077

- Minor correction for 3.076 where Setup Capture now has a space in the filename for better readability

Version 3.076

- Fixed program in 3.066 where virtual file deletes would not work when located in shell folders
- Fixed issue where Textpad 5.0 showed plus signs for normal files
- Setup Capture & log monitor use XP manifest and look nicer on XP
- Update for CreateProcess to set correct error code on failure
- Fixed issues with DWG Trueview 2007
- Fixed installer error "0xabba0205" which resulted if old files were left behind

Version 3.070

- Added "InventoryName" to `package.ini` (Chapter 8, "Package.ini format," on page 95), this will be used by desktop management and license control systems. Default value is the same as the sandbox name.

Beta

- Added ChildProcessEnvironmentDefault and ChildProcessEnvironmentExceptions to `package.ini`, allowing you to control what happens to child processes (run as system or virtual)
- Vregtool updated to support exporting `.tvr` registry files back to text format. Virtual registry text files can also mark registry subkeys and values as being deleted.
- Environment variable changes were accidently being excluded from snapshots in previous builds, this is now fixed.

Version 3.067, Mar 13 2007

- Fixed an issue where application may lock up if application threads do not quit and a force-kill occurs
- Fixed an issue where logging will lock-up if an application probes a registry key with more than 128K of data

Version 3.066, Mar 12 2007

- Virtual Registry is now scanned for corruption on startup, if detected a backup copy is used. We found registry can become corrupted on USB flash devices where the app was plugged in and out without "safe remove"
- SetupCapture with add a commented-out section to run Internet Explorer for using IE plugins
- Fixed an issue where Thinstalled Internet Explorer would crash when McAfee Enterprise Anti-virus script blocker was installed
- Added `package.ini` option VirtualizeExternalOutOfProcessCOM (default is 1). This allows you execute all Out-of-process COM objects on the system without being loaded into the virtual environment
- Now you can determine which version of VMware Project North Star (Thinstall) an application has been packaged by running with "package.exe -ThinstallVersion"
- SetupCapture displays status messages in main window instead of seperate DOS window
- Fix to enable NT4
- Update to services, some support for shared process services
- Allowed access to device drivers referenced through `\\?\` format
- Setup Capture no longer scans HKEY_CURRENT_CONFIG, this subtree is not needed for applications
- Fixed issue where Vista native File/Open dialogs crash under VMware Project North Star (Thinstall)
- Added support for `LoadCursorFromFile`

Version 3.052, Feb 9 2007

- Virtual Registry is now scanned for corruption on startup, if detected a backup copy is used. We found registry can become corrupted on USB flash devices where the app was plugged in and out without "safe remove"
- SetupCapture with add a commented-out section to run Internet Explorer for using IE plugins
- Fixed an issue where Thinstalled Internet Explorer would crash when McAfee Enterprise Anti-virus script blocker was installed
- Added `package.ini` option VirtualizeExternalOutOfProcessCOM (default is 1). This allows you execute all Out-of-process COM objects on the system without being loaded into the virtual environment
- Now you can determine which version of VMware Project North Star (Thinstall) an application has been packaged by running with "package.exe -ThinstallVersion"

Version 3.051, Feb 9 2007

- Fixed problem where message "Could not verify Thinstall License" would appear on startup when running from Guest Accounts and sub processes
- Fixed problem where File/Open dialog would not show files for Desktop and My Documents on Vista

Beta

- Fixed problem where `Helppane.exe` was being launched on process shutdown for MFC apps on Vista
- By default network mapped drives and removable disk will not be sandboxed/isolated, you can change this with an option in `package.ini` (Chapter 8, "Package.ini format," on page 95)
- Fixed problem with out-of-process single-use COM objects, 2nd instances were not being registered
- Added `package.ini` option to specify specific COM object IDs which should be created on the system instead of virtual environment

Version 3.043, Jan 30 2007

- Fixed "ExecuteVirtualProcess", added documentation for script API "ExpandPath" ("ExpandPath" on page 122)
- Added Script function WaitForProcess ("WaitForProcess" on page 128)
- ExecuteVirtualProcess and ExecuteExternalProcess will now return a process ID
- Updated SetupCapture so `##Attributes.ini` is stripped out for `%Desktop%` and `%Personal%` so subdirectories are not redirected to sandbox
- Fixed problem with past release where `%Desktop%` and `%Personal%` always used default isolation level instead of specified level
- Fixed problem where applications could not access COM and LPT ports
- Fixed problem with ActivePerl where probes for virtual file security info would fail
- Fixed problem for TeamMate Audit Management System
- Setupcapture will now exclude `HKEY_LOCAL_MACHINE\system\CurrentControlSet\`control from captures (was capturing computername)
- Fixed Data Execution Prevention problem for Office products that occurs on w2k3 when DEP is enabled
- Fixed problem with Acrobat Reader 8 where Last Access Time for parent directories was incorrect
- Changed virtual services so that the WMI service can be accessed by virtualized applications

Version 3.035

- Fixed problem for 64bit Windows XP for EXE that support addresses above 2GB.
- Added Scripting support. (Chapter 9, "Scripting," on page 117)
- Fixed problem with Office XP activation system running on Windows 2003
- Setup Capture will now scan other drives besides just drive `c:`

Version 3.032

- Fixed problem where "My Pictures" and "My Music" would be placed in sandbox instead of on host PC by default
- Fixed problem with registered out-of-process COM objects being using as in-process objects
- Added support for executing .cmd files

Version 3.031

- Fixed Installer Package for Windows Vista
- Generated EXE and Shortcut EXEs are smaller because we only include the first icon, you can force all icons to be included ("RetainAllIcons" on page 111)
- You can now specify a `.ico` file for an application icon in addition to an `.EXE` file. You can also use NULL for no icon. ("Icon" on page 103)
- Optimized Registry access—applications that use a lot of registry keys will start faster.
- Fixed problem where Outlook 2003 crashes on exit

Beta

- Fixed startup issue for Outlook XP Edition
- Added better support for handling application with a large number of files, vftool should comfortably handle applications with 100,000+ files
- Fix where Skype would get stuck in endless loop during connection

Version 3.023

- LogMonitor File/Open dialog now shows `*.trace` files
- Fixed `package.ini` option "WorkingDirectory=", SetupCapture will automatically add this entry for shortcuts which specify a working directory
- Fixed registry bug where Skype did not work, may fix some network apps as well. This change required a virtual registry format change, so previously deployed apps will lose their Sandbox data if you upgrade the EXE and don't change the sandbox name.
- Fixed problem where App Paths should be inherited by child processes, fixes Gimp
- Added work-around for Symantec AV where it would scan the entire EXE from a network drive for any app that uses networking. Now apps should start up quickly from network shares when SAV is installed.
- Fixed problem where status bar never disappears when parent process does not create any windows
- Licensed versions of VMware Project North Star (Thinstall) are now available, if you have purchased VMware Project North Star (Thinstall) VS the web server will automatically provide you with a licensed download. If you have existing projects, all that needs to done is rebuild the apps using this new licensed version and they will not time out. The applications should display a Thinstall statusbar message saying something like "Licensed to Company XYZ" during startup.
- End user license agreement for VMware Project North Star (Thinstall) was added and is displayed for applications that contain the VMware Project North Star (Thinstall) VOS on first execution by a new user. If you have a license agreement, this can be turned off if you accept the EULA on behalf of your users, and you are deploying only internally within your company or otherwise have the legal right to accept the EULA terms for your users.
- Fixed problem with Office XP, where it might crash during Activation
- Fixed problem where SetupCapture did not continue after reboot (hopefully we really got it this time!)
- Fixed problem where winamp would not remember .ini setting changes if an .ini was included in the original package
- Added Work-around for running on systems with TestComplete installed, TC has a bug that will sporadically hang processes with multiple threads on shutdown.
- Added some COM related logging
- Fixed problem with SafeCast work-around that could crash File/Open dialog on Windows 2000

Version 3.017

- Fixed problem where File/Open dialog may crash from 3.015
- Fixed problem for Trillian
- Fixed problem where some captured MSI packages on w2k did not work on xp (outlook 2003), you will need to recapture the package to get this fixed

Version 3.015

- SetupCapture will now include environment variables in the capture, the PATH variable is treated special (Chapter 12, "Application PATH and Environment Variables," on page 133)
- Fixed bug where many .NET 2.0 would not work, there are not any known .NET 2.0 issues now
- Fixed bug in services which were causing some apps built with 3.010 to crash

Beta

- Improved services support: DeleteService, EnumServices, EnumDependentServices, etc.
- CreateProcess now supports `.bat` files
- Improved out-of-process COM for Vista
- Fixed problem where registry last-modified time stamps may return invalid dates
- Fixed OpenFile problem for Adobe Acrobat 4.0
- Log monitor will now exclude more entries from the potential errors section which are not actually errors
- Added compatibility for running FlexLM and SafeDisc

Version 3.010

- Added initial support for virtual services
- Added support services based COM objects
- Added support for Active Directory authentication (Chapter 11, "Access Control using Active Directory," on page 131)
- Fixed problem where installing an application update (overwriting the original EXE) may prevent the app from running again
- Added "Icon=" option on `package.ini` to allow you to specify an alternate icon. Currently this is limited to pointing to an EXE, but later we'll add support for .ico files
- Fixed problems where some COM creation calls failed on Windows Vista
- Added status update print for tlink for large builds
- Turned on compression of VOS, so it takes 340k on disk instead of 1MB.
- We've added a blog (https://thinstall.com/account/index.php) and discussion forum (http://thinstall.com/thintalk) for beta users

Version 3.007

- Fixed problem where some apps would crash and consume 100% CPU on exit
- Fixed problem where setup capture would not continue after reboot during the capture process
- Active Directory integration, environment variables, and command-line arguments didn't make it into this release we are still working out a few problems and were occupied with Citrix iForum this week. These will likely be in next Friday.
- Updated documentation about controlling sandbox locations (Chapter 5, "Sandbox Overview," on page 65).

Version 3.003

- Fixed problem where applications did not run on Windows Vista
- Added new text registry output format for setup capture. This format is fast to generate and process as well as easy to edit and review.
- Added support for Outlook 2003 (may crash & chew CPU on quit)
- Added support Oracle 10g client (Oracle 10g still needs better shortcut EXE generation)
- Added sandbox locking mechanism to prevent problems where 2 computers try to write to the same virtual registry over a network simultaneously, if multiple computers try to create sandboxes at the same location they will create a unique directory with their computer name
- Made %Desktop% Merged Isolation (Chapter 4, "Isolation Modes," on page 61) by default—allowing users to save files to their desktop (previously went to desktop)
- Added support for compressed packages (still needs compression for VOS & Virtual Registry)

Beta

Getting Started

2

This walk-thru assumes you have installed VMware Project North Star (Thinstall) to your development PC using the .MSI provided in your online account.

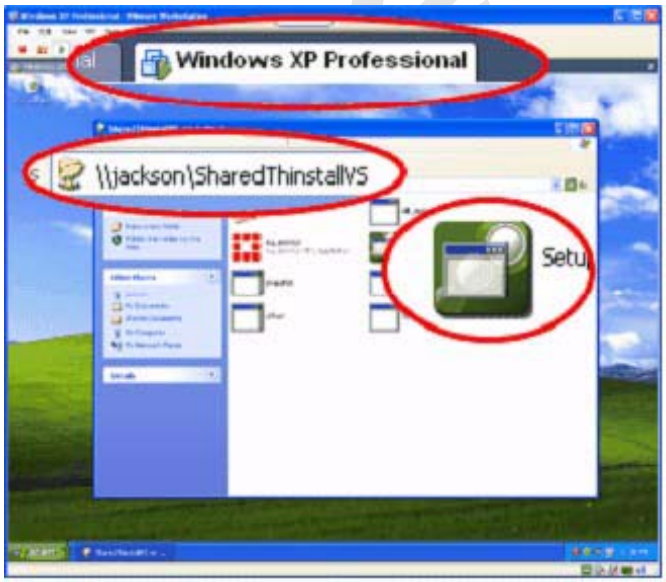
1 Setup a clean PC if you do not already have one.

"Using a Clean PC for Capturing" on page 41

"Installing VM Server" on page 41

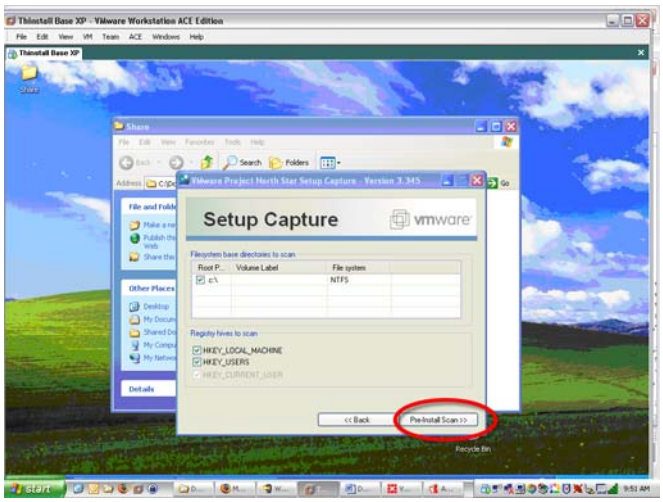
2 Share your `ThinstalVS` directory and launch SetupCapture on a clean PC

We recommend launching Setup Capture directly from a network share to ensure no changes are made to the local PC except by the installed application. In order to create a full capture of an application you should have available a clean Windows system for the lowest platform you need to support. For example, use Windows 2000 if you plan to support 2k,XP,2k3, and Vista.



3 Use default options and select “Pre-install Scan”

On a fast clean PC, the scan process should take ~10 seconds for Windows XP.

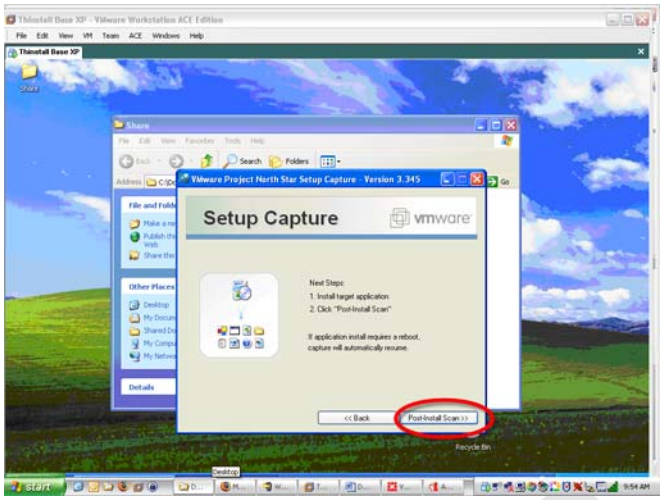


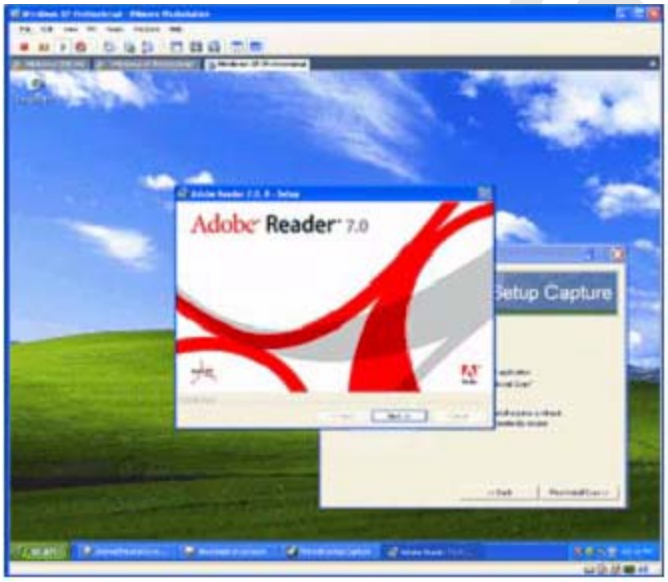
4 When the scan finishes, install your target application.

After installing your application you may also want to configure it as you want it to appear when the resulting Thinstalled application is first executed by a new user.

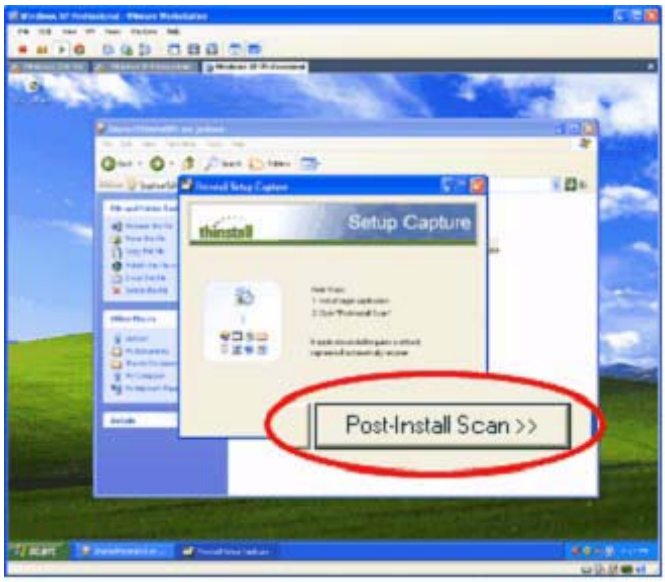
Beta

If the application does not have an installer, you can manually copy the application files to the capture machine and add desktop or shortmenu shortcuts to EXE files you want to generate VMware Project North Star (Thinstall) EXEs for during the build process. Setup capture will automatically detect all the changes made to the guest machine including installation of 3rd party libraries, COM registrations, and registry edits.



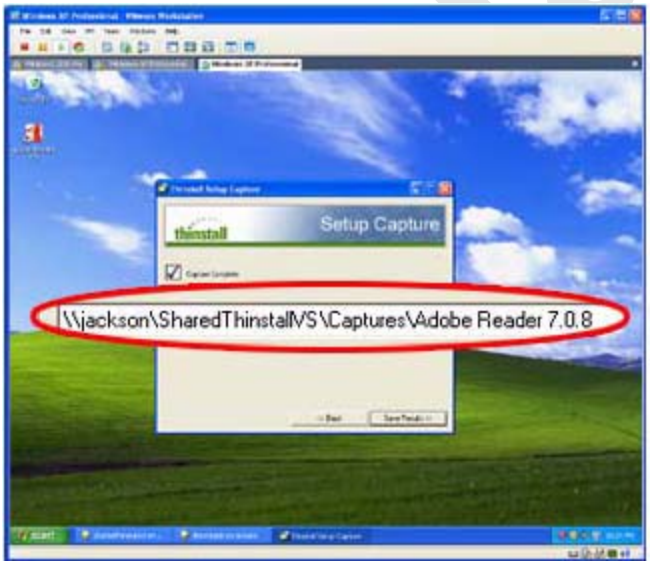


5 After the application has been installed and configured, click **Post-install Scan**.



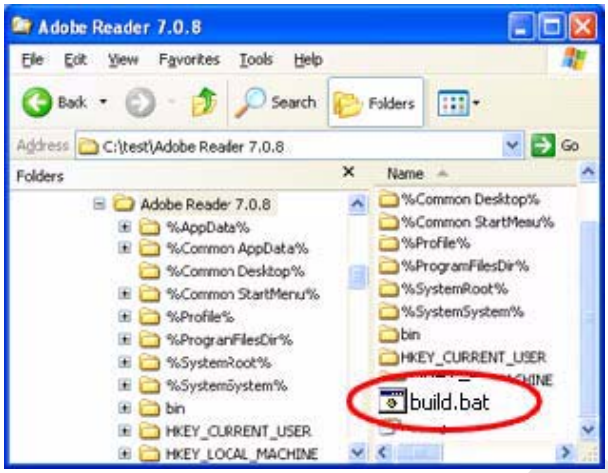
6 Confirm Location to save project and Click **Save Results >>**.

VMware Project North Star (Thinstall) will suggest a name for the project. If the application installed multiple components, you may need to adjust the suggested name.

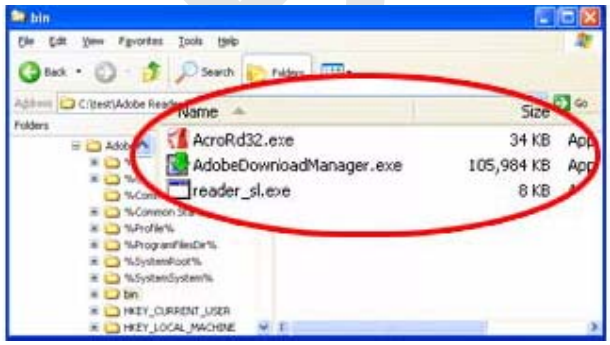


7 Switch to dev PC with VMware Project North Star (Thinstall)VS installed, and execute `build.bat`.

This will execute the virtual registry compiler, virtual filesystem packager, and runtime linker to produce one or more EXE files in the `bin` directory under your project. If you installed VMware Project North Star (Thinstall)VS to a location other `c:\Program Files\ThinstallVS`, you need to set the environment variable THINSTALL_BIN before executing the `build.bat` file. VMware Project North Star (Thinstall)VS can be run from a shared network location so it does not need to installed locally.



NOTE VMware Project North Star (Thinstall) selects one "primary" EXE to host the VMware Project North Star (Thinstall) runtime plus application registry data and files. The other EXE files are simply "Shortcuts" to the main EXE and can be deleted prior to distribution to users if they are not needed. EXEs are selected based on which shortcuts were installed by the application during install, so you can delete unimportant shortcuts before completing the post-install setupcapture scan on capture machine. If you want a different primary EXE after SetupCapture has completed, you can edit the package`.ini` file. Shortcut EXEs cannot be run unless the "primary" EXE is located in the same directory. Shortcut and the primary EXE will share a common virtual registry and filesystem and can interact with each other.



Editing package.ini

For example, with Acrobat reader we can make `AcroRd32.exe` the only EXE by editing:

- Before edit:

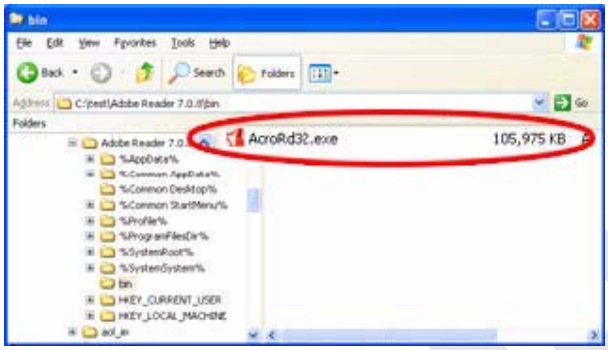
```
[BuildOptions]
OutDir=bin
SandboxName=reader7

[AdobeDownloadManager.exe]
Source=%ProgramFilesDir%\Common
Files\Adobe\ESD\AdobeDownloadManager.exe
ReadOnlyData=bin\Package.ro.tvr
```

Beta

```
[AcroRd32.exe]
Source=%ProgramFilesDir%\Adobe\Acrobat 7.0\Reader\AcroRd32.exe
Shortcut=AdobeDownloadManager.exe

[reader_sl.exe]
Source=%ProgramFilesDir%\Adobe\Acrobat 7.0\Reader\reader_sl.exe
Shortcut=AdobeDownloadManager.exe
```

- After edit:

```
[BuildOptions]
OutDir=bin
SandboxName=reader7

[AcroRd32.exe]
Source=%ProgramFilesDir%\Adobe\Acrobat 7.0\Reader\AcroRd32.exe
ReadOnlyData=bin\Package.ro.tvr
```



At this point you have a virtualized application which can be run directly from a network share on restricted accounts without installation or modification to local PCs. The Thinstalled EXE is read-only and never changes, so it can be placed in a central location where it is shared by multiple users. VMware Project North Star (Thinstall) supports files of unlimited size and provides transparent streaming capabilities so a 2 terra-byte file can be launched over the LAN in seconds, only the data required to launch the application is pulled across the network. Windows 2k and XP have a known issue where they will not display icons in the Explorer shell for very large EXE files, the exact size depends on the local PC, but typically around 500MB-1GB. Because shortcut EXEs are always very small, they will never encounter this problem, only the primary EXE and a dummy primary EXE can be used as a work-around for large packages.

Controlling the location of the Sandbox

The sandbox is the directory where all changes made by the application will be stored. The next time the user executes the application, those changes will be remembered from the sandbox. By deleting the sandbox directory, you can instantly revert the application back to it's captured state.

By default the sandbox is located in the users `%AppData%` directory, i.e.:

```
c:\documents and settings\username\Application Data\Thinstall
```

Using Active Directory, the user's %AppData% is often mapped to a shared network drive to allow for easy backups. When this is the case, users can log in on any machine and retain their application settings. VMware Project North Star (Thinstall) transparently remaps registry and file data while the application is running to enable shared application profile information to instantly migrate to different OSes. For example, if the application registers DLLs to `c:\winnt\system32` while running on w2k the user can quit the application and login on an XP box. On the XP box, the files will automatically appear to exist at `c:\windows\system32` and all related registry keys will automatically point to `c:\windows\system32`. On Windows Vista, VMware Project North Star (Thinstall) automatically moves Windows SXS (Side by Side) DLLs and policy information to match Vista versus XP file path styles. This functionally allows most applications to instantly migrate to newer or older operating systems. VMware Project North Star (Thinstall) provides SXS support for applications running on Windows 2000 even though the underlying operating system does not, this enables most applications captured on XP to run on w2k without changes.

Beta

NOTE Only one computer may be actively using a shared sandbox simultaneously. If a sandbox is already in use by another computer VMware Project North Star (Thinstall) will display a warning and create a new sandbox to allow the user to continue working until the previous copy closes.

Portable Applications

To make an application portable so that it can be executed from a USB Flash device, IPod, etc. the sandbox should operate in "local" mode. In this mode, the sandbox is in a subdirectory relative to the EXE. This mode is not recommend for shared environments like Terminal Server unless each user has their own copy of the application because sandboxes are not user specific but instead become location specific. To cause an application to run with a local sandbox, simply create a directory called `Thinstall` in the same directory as your Thinstalled application. You can also move the `Thinstall` directory from `%AppData%` to the application directory to take existing application settings and make them operate in local mode.

Using a Clean PC for Capturing

VMware Project North Star's Setup Capture application works by taking 2 snapshots for a PC's filesystem and registry before and after installation the target application. The VMware Project North Star (Thinstall) project is created using the differences between the two snapshots. We recommend using a clean PC to capture application installs because installers will skip files that already exist on the PC. If the installer skips files, they will not be included in the VMware Project North Star (Thinstall) project and the application may fail to run on other PCs where the files do not exist. Another positive aspect of using a clean PC is that SetupCapture can scan the PCs filesystem and registry very quickly when it is clean, typically in 15 seconds for a clean PC, but may take several minutes to an hour for non-clean PCs.

What is a clean PC?

A clean PC is a PC which has Windows base OS installed and nothing else. In a corporate environment where you have a base desktop image, the base desktop image is your "clean" PC. In such cases the back desktop PC may already have some components and libraries installed (for example .NET 2.0). When you capture a package on a PC which has .NET 2.0 already installed, .NET 2.0 will not be included in your package and the application will then only run on PCs which have .NET 2.0 already installed. We have a guide for installing VMWare Server (a free product) ("Installing VM Server" on page 41) which can provide you with a clean environment any time you need it for capturing and testing purposes.

What version of Windows should I use for capturing?

We recommend capturing on the lowest platform you plan to support for your users. In most cases this will be WIndows 2000 or Windows XP. Most packages captured on XP will work on 2k, however in some cases Windows XP comes with some DLLs which Windows 2k does not have and they will be excluded from the package if the application normally installs these DLLs.

I'm a developer and my application changes every time I rebuild, do I need to recapture it every time?

No. Once you have created the VMware Project North Star (Thinstall) project structure, you can overwrite files in the project with newer versions and then rebuild the application without the capture process. The process of compiling the application, copying updated files, and building a VMware Project North Star (Thinstall) EXE can be easily automated using batch files.

Installing VM Server

When creating a VMware Project North Star (Thinstall) project using SetupCapture, we recommend using a "clean" install of Windows. If you don't already have a clean PC sitting around for this purpose, we recommend using a product like VMWare to maintain a clean image of Windows.

With VMWare, you can install Windows into a virtual machine one time and take a "snapshot" of the entire machine in its clean state (no application installed). Then after you have used VMware Project North Star's Setup Capture application to capture an application's install, you can use VMWare to discard your changes and revert the virtual machine to a clean state where it is ready to capture a new application.

Beta

The guide below shows how to install and configure VMWare Server with a base clean OS. As of this writing, VMWare Server is available for free from VMWare. MSDN subscribers can also obtain a free license to a similar product from Microsoft called VirtualPC. We recommend VMWare over VirtualPC if you have a choice in selection.

Step 1 Go to the VMWare web site.

At http://www.VMWare.com you can register for a license, download the installation package, and access documentation for VMWare.

You can find a lot of helpful information about how to work with VMWare here.

Step 2 Register for a VMWare Server license.

The first thing you will need to do is to register for a VMWare Server license. Yes, VMWare Server is free, but you still need a license for it.

Go to http://register.VMWare.com/content/registration.html to register.

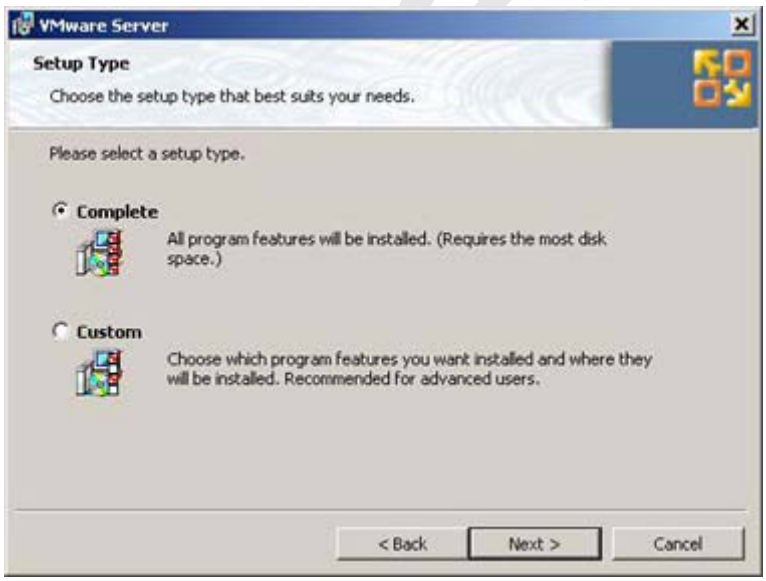
Make sure to retain the serial number provided at the bottom of the web page, after submitting your registration, since this is the only time it is provided.

Step 3 Download the VMWare Server installer.

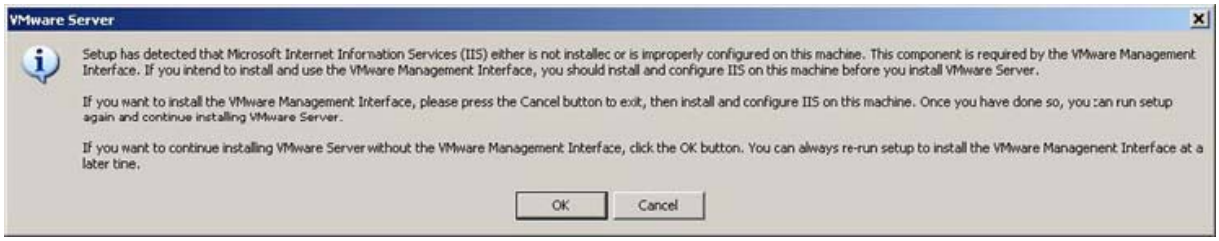
VMWare Server is available for free download at http://www.VMWare.com/download/server.

Step 4 Install VMWare Server.

Double-click on the installer. The installation steps are straight forward. Just follow the on-screen instructions selecting all defaults.


VMware Server
Setup Type
Choose the setup type that best suits your needs.
Please select a setup type.
Complete
All program features will be installed. (Requires the most disk space.)
Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.
< Back
Next >
Cancel

If the installer warns you about "Microsoft Internet Information Services (IIS) not installed ...", click **OK** to continue.


VMware Server
Setup has detected that Microsoft Internet Information Services (IIS) either is not installec or is improperly configured on this machine. This component is required by the VMware Management Interface. If you intend to install and use the VMware Management Interface, you should install and configure IIS on this machine before you install VMware Server.
If you want to install the VMware Management Interface, please press the Cancel button to exit, then install and configure IIS on this machine. Once you have done so, you can run setup again and continue installing VMware Server.
If you want to continue installing VMware Server without the VMware Management Interface, click the OK button. You can always re-run setup to install the VMware Management Interface at a later time.
OK
Cancel

If the installer warns you about CD/DVD autorun being enabled, check the disable box.

When the installer reaches the Customer Information panel, enter the serial number, obtained earlier, in the Serial Number field.

(You may skip this step and enter the serial number later on when you want to use VMWare Server.)

You may have to restart to complete the installation.

Step 5 Launch the VMWare Server Console.

Launch the VMWare console program with the "VMWare Server Console" icon that was placed on the desktop.

Alternately use the menu: **Start>All Programs>VMWare>VMWare Server>VMWare Server Console**.

In the "Connect to Host" dialog, select the **Local** radio button and click **Ok**.

VMware Server Console - Connect to Host
VMware Server Console
Select the VMware host that you want to connect to.
To access virtual machines on the local computer you are using, select Local host. To access virtual machines on a networked host, select Remote host and enter the host name and a valid user name and password.
Local host
Remote host
Host name:
User name:
Password:
OK
Cancel

You now should have the VMWare Server Console running, with no virtual machines open.

Step 6 Create a new virtual machine.

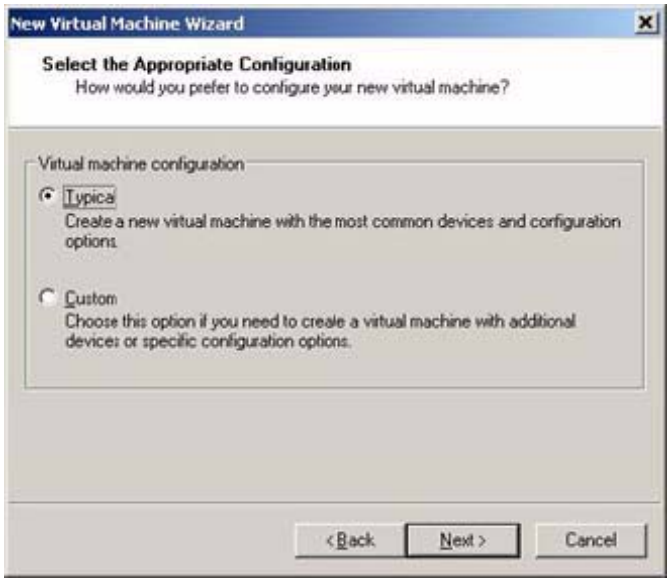
Click **New Virtual Machine** to bring up the "New Virtual Machine Wizard".

Alternately use the menu: **File>New>Virtual Machine...** menu item.

Click **Next**.

Select a **Typical** virtual machine configuration.

Click **Next**.


New Virtual Machine Wizard
Select the Appropriate Configuration
How would you prefer to configure your new virtual machine?
Virtual machine configuration
Typical
Create a new virtual machine with the most common devices and configuration options.
Custom
Choose this option if you need to create a virtual machine with additional devices or specific configuration options.
< Back
Next >
Cancel

Select the operating system that you will be installing on the virtual machine.

This should be the earliest operating system that you want to run your thinstalled application on.

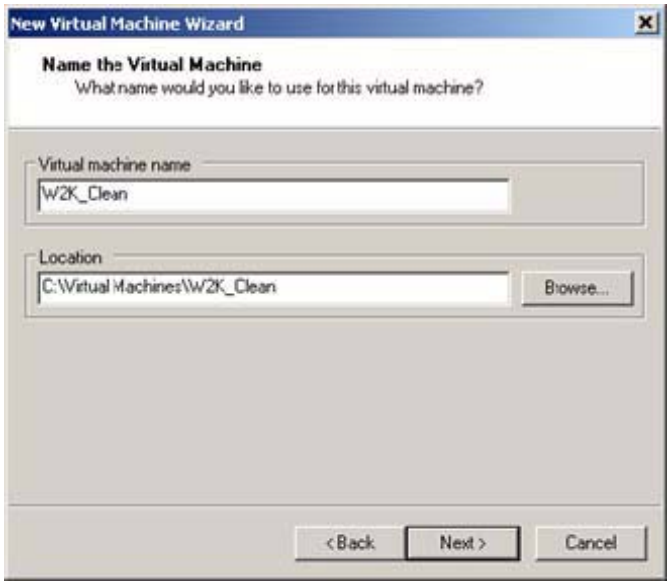
In this example Windows 2000 professional will be used. This will allow a thinstalled application to run on W2K,WXP,W2K3, and Vista.

Select **Microsoft Windows** and **Windows 2000 Professional**

Click **Next**.

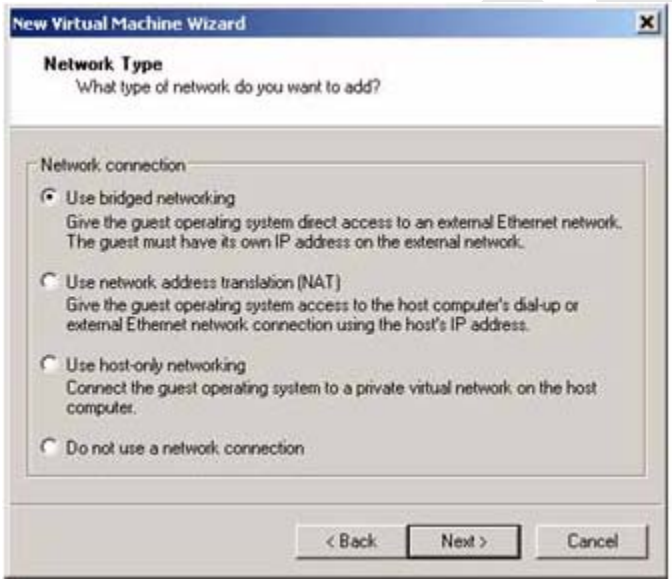
Enter a display name and disk location for the virtual machine directory, such as `Win2K_Clean`.

Click **Next**.



Select **Use bridged networking**, this will allow the virtual machine to communicate with other PCs on your LAN.

Click **Next**.

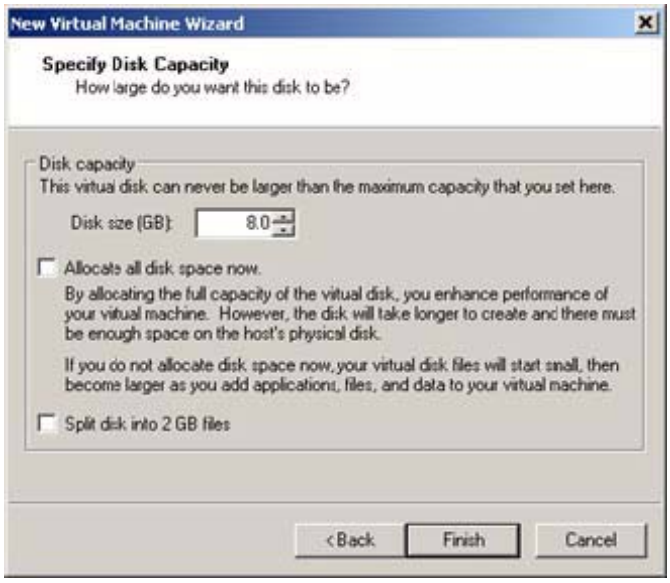

New Virtual Machine Wizard
Network Type
What type of network do you want to add?
Network connection
Use bridged networking
Give the guest operating system direct access to an external Ethernet network. The guest must have its own IP address on the external network.
Use network address translation (NAT)
Give the guest operating system access to the host computer's dial-up or external Ethernet network connection using the host's IP address.
Use host-only networking
Connect the guest operating system to a private virtual network on the host computer.
Do not use a network connection
< Back
Next >
Cancel

The default of 8.0GB is adequate for most uses.

If you plan to capture large application installs, you may want to increase this size to 20GB.

Uncheck to **Allocate all disk space now**.

Click **Finish**.


New Virtual Machine Wizard
Specify Disk Capacity
How large do you want this disk to be?
Disk capacity
This virtual disk can never be larger than the maximum capacity that you set here.
Disk size (GB):
8.0
Allocate all disk space now.
By allocating the full capacity of the virtual disk, you enhance performance of your virtual machine. However, the disk will take longer to create and there must be enough space on the host's physical disk.
If you do not allocate disk space now, your virtual disk files will start small, then become larger as you add applications, files, and data to your virtual machine.
Split disk into 2 GB files
< Back
Finish
Cancel

A directory containing several files is created at the location you specified earlier. (Defaults to `C:\Virtual Machines`).

If you examine the contents of this directory you will find two files of interest. The machine configuration file, `Windows 2000 Professional.vmx`, may be doubled clicked to launch and open VMWare Server Console. Another file of interest is the virtual disk file, `Windows 2000 Professional.vmdk`, which holds the contents of the hard drive. This file will grow as need to hold the contents of the hard drive, up to the limit specified above.

You should now have a fresh new virtual machine.

Step 7 Configure the virtual machine to boot from the CD drive.

A virtual machine is created with a completely empty hard drive and is not able to boot up.

You will need to configure the virtual machine to boot from your installation CD, either physical or ISO file.

Select the **VM>Settings...** menu to edit the virtual machine settings.

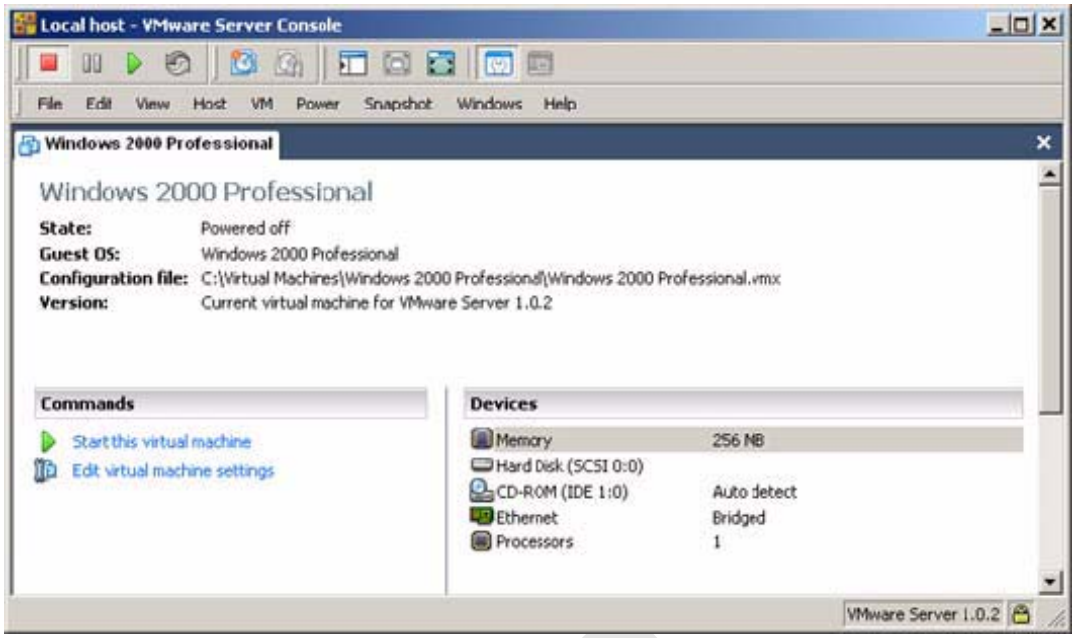
Select the CD-ROM device.

Click once to place a check mark in the **connect at power on** check box.

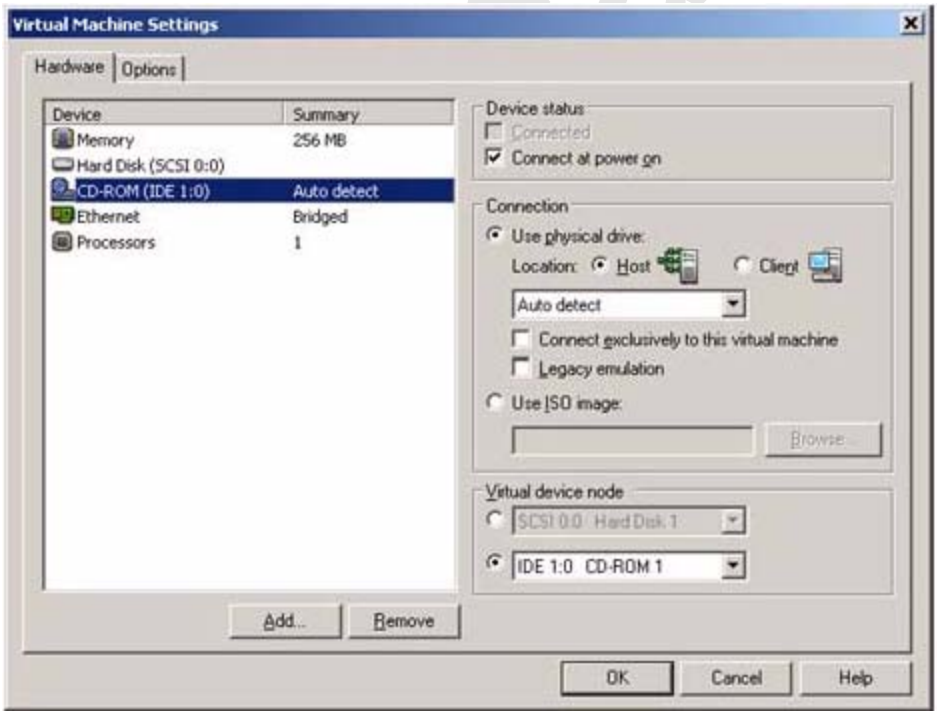
If you have a physical installation CD: select the **use physical drive** radio button, select the **Host** radio button, and select **Auto detect** menu item.

Beta

Insert the CD into your physical machine's CD drive.


Local host - VMware Server Console
File Edit View Host VM Power Snapshot Windows Help
Windows 2000 Professional
Windows 2000 Professional
State: Powered off
Guest OS: Windows 2000 Professional
Configuration file: C:\Virtual Machines\Windows 2000 Professional\Windows 2000 Professional.vmx
Version: Current virtual machine for VMware Server 1.0.2
Commands
Start this virtual machine
Edit virtual machine settings
Devices
Memory 256 MB
Hard Disk (SCSI 0:0)
CD-ROM (IDE 1:0) Auto detect
Ethernet Bridged
Processors 1
VMware Server 1.0.2

If you will be using an ISO file: select the **use ISO image** radio button, browse to the location of the file.

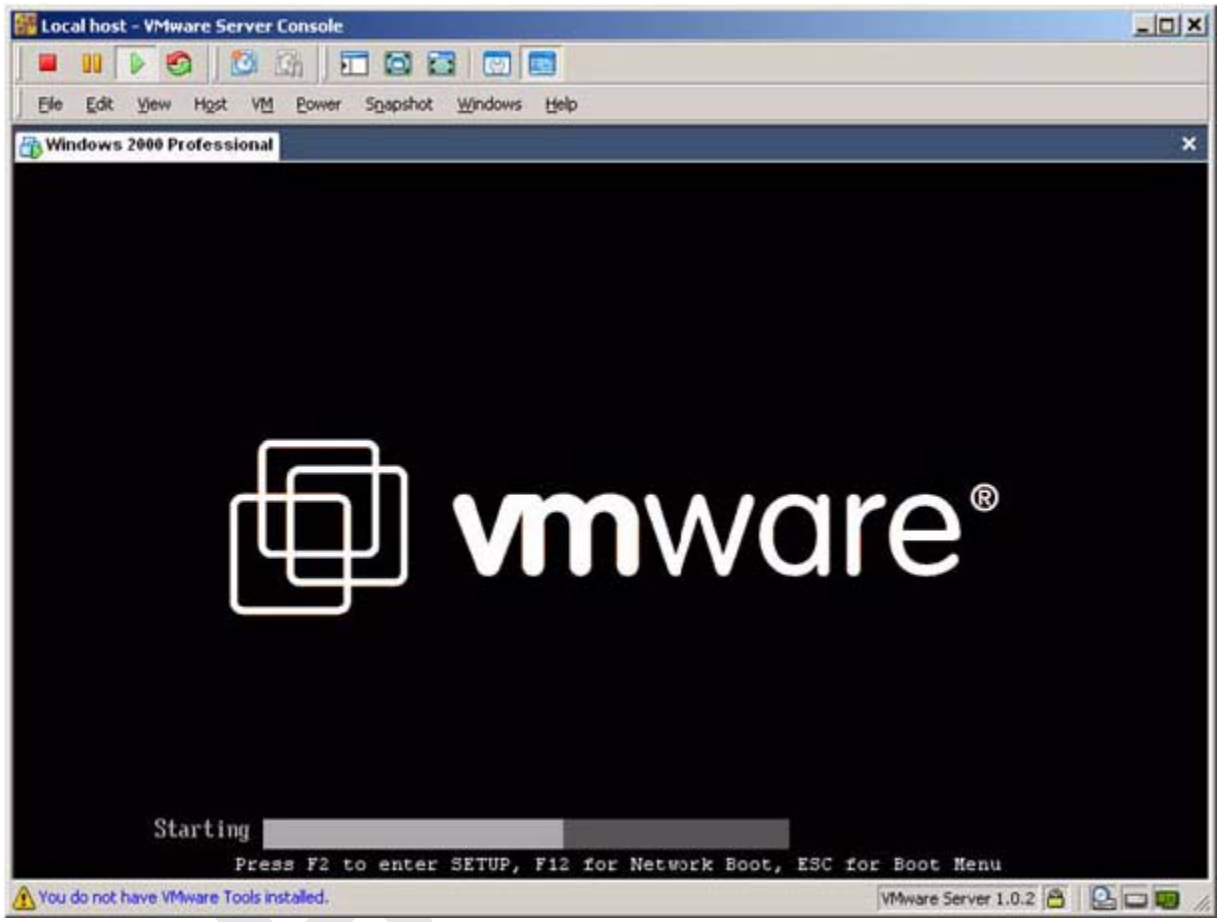


Click the **Ok** to close the Machine Setting dialog.

Step 8 Start up the virtual machine and begin installing the operating system.

Select the **Power>Power On** menu to start the virtual machine running.

At this point you will see the virtual machine's desk top turn black, then the BIOS post screen, and later the installer startup screen will appear.


Local host - VMware Server Console
File Edit View Host VM Power Snapshot Windows Help
Windows 2000 Professional
vmware®
Starting
Press F2 to enter SETUP, F12 for Network Boot, ESC for Boot Menu
You do not have VMware Tools installed.
VMware Server 1.0.2

To interact with the virtual machine you need to click in the virtual machine's desk top. This will cause the virtual machine to 'capture' the keyboard and mouse as input devices. All further keystrokes and mouse clicks will be delivered to the virtual machine, not to your physical machine. To release the keyboard and mouse at any time press the Control and Atl keys simultaneously.

If you see a 'No operating system found' message or 'No bootable CD' dialog, then you need to Power Off the virtual machine (**Power>Power off**) and change the CD configuration or try a different installation CD.

Windows 2000 Professional - Virtual Machine
No bootable CD, floppy or hard disk was detected.
To install an operating system, insert a bootable CD or floppy and restart the virtual machine by clicking the Reset button.
OK

Make sure the **connect at power on** check box is selected or try the **legacy emulation** check box (see step 7).

If all goes well you should see the Windows installer appear and start asking you to make decisions.

Follow the on screen instructions to install the operating system you have selected. Typically this involves creating a new disk partition, formatting this partition, providing replies to the installation program, and waiting around for the installation to complete. When in doubt use the default settings suggested by the installer.

Usually the installation will finish by restarting the machine. Once the machine has restarted, resist the temptation to use the machine or change any settings. You want to preserve at least one copy of the machine just after installing.

Step 9 Install VMWare Tools

VMWare Tools is a set of drivers and services that allow the virtual and physical machines to interact smoothly together.

Select the **VM>Install VMWare tools** menu to start the installation of VMWare tools.

This will temporarily connect a CD drive to the virtual machine with an installation CD. For Windows 2000, the installer automatically launches on the virtual machine. If it does not appear, navigate to the new CD drive, locate and launch the VMWare Tools installer (something like `D:\setup.exe`). Accept all defaults from the installer and restart the virtual machine.

Once installed VMWare tools allows you to size the monitor to your liking, drag/drop and copy/paste between the virtual machine and physical machine.

Step 10 Note: This step is optional, it will cause your virtual machine to revert back to the “clean” state each time it is powered off.

Make the hard drive ‘Independent-Nonpersistent’

Use the virtual machine’s **Start>Log Off** menu to shutdown the virtual machine.

Once the virtual machine has shutdown, Select the **VM>Settings...** menu item and select the Hard Disk device.

Click the **Advanced** button.

Click once to place a check mark in the **Independent** check box.

Select the **Nonpersistent** radio button.

Click **Ok** to close the Advanced dialog.

Click **Ok** to close the Virtual Machine Settings dialog.

Changing the hard drive to ‘Independent-Nonpersistent’ causes all changes to the virtual machine’s hard drive go to a separate temporary virtual disk file. When the machine is powered off the file is discarded. The benefit of this is the ability to always return to a clean machine state by simply power cycling the machine. Additionally you power off the virtual machine, rather than wait for shutting down, without triggering a disk scan (or corruption) on the next boot.

Step 11 Preserve your clean virtual machine.

Exit VMWare Server Console and create a backup copy of the virtual machine directory. This will preserve the virtual machine at its clean state. It is very easy to contaminate or otherwise break a virtual machine, sometimes without even knowing it. If you have a back up copy you won't have to wait through the windows installation once again.

Step 12 Adjust the virtual machine to suit your environment.

You may need to change some setting to fit your work place environment, such as network connection settings. Remember to keep the virtual machine in 'Independent-Nonpersistent' mode until you know the changes you are make work. Then Power down the machine, remove the 'Independent-Nonpersistentmode', Power up, make the changes, Shut down the machine, and return to 'Independent-Nonpersistent'. This technique will minimize the impact of using the machine while not in 'Independent-Nonpersistent' mode. Remember to use **Start>Log Off**, (not Power off) while not in 'Independent-Nonpersistent' mode, since this may trigger a disk scan upon the next boot.

Beta

How VMware Project North Star (Thinstall) Works

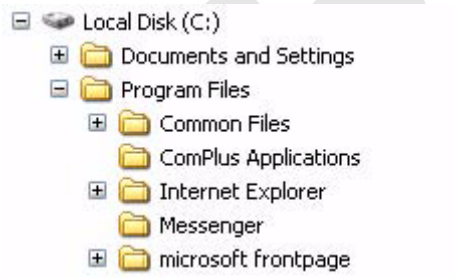
3

VMware Project North Star (Thinstall) works by using a build process to "link" the Virtual Operating System (VOS) with a compressed embedded filesystem and registry into a single EXE file. The EXE file can run with zero installation, and without decompressing files to disk, from any data source including a user's desktop, a network path, or removable storage like USB Flash and CDROM. VMware Project North Star (Thinstall) enables applications to run directly from slower speed storage devices such as USB flash or network shares in an efficient manner by using block-based streaming with transparent decompression. ("Block based Streaming" on page 53)

VMware Project North Star (Thinstall) accomplishes "Zero Installation" by presenting a virtual environment to the running application, making it appear as if all of its files, registry entries, environment variables, COM/ActiveX controls, services, etc. have already been installed on the PC, even though, in reality, no changes have been made.

The virtual environment presented to the application is a merged view of files installed by the application and files already existing on the PC. For example, consider a host PC which has a filesystem that looks like this:

Filesystem as seen by Windows Explorer



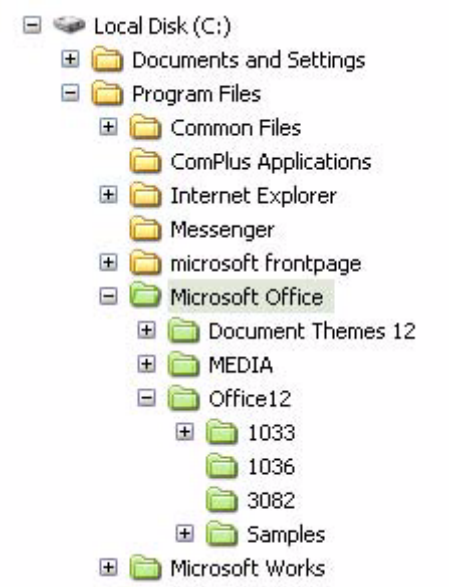
Microsoft Office creates various directories during it's installation process, including:

```
C:\Program files\Microsoft Office\...
C:\Program files\Microsoft Works\...
```

When running a "thinstalled" version of Microsoft Office, the application would see all of the original files on the PC plus the additional directories installed by Microsoft Office. If the user used the File/Open dialog, he or she would see the following:

Beta

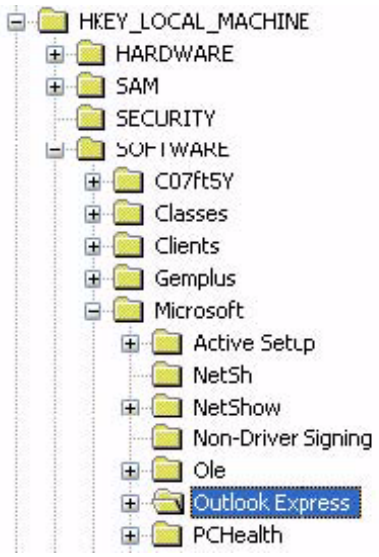
Filesystem as seen by Thinstalled Office



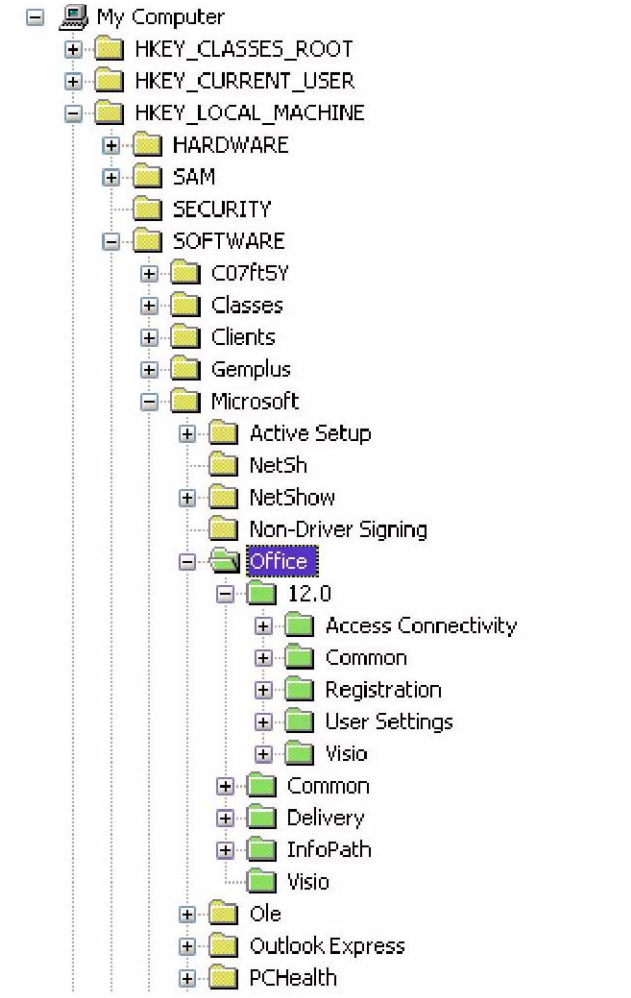
Because these directories are not actually created on the host PC, the PC remains unchanged and thinstalled applications will not negatively impact other applications on the same PC.

VMware Project North Star (Thinstall) presents a merged view of the system registry as well:

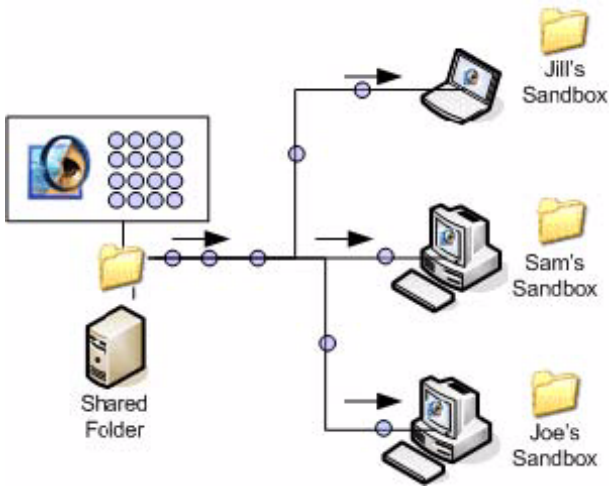
Registry as seen by Windows Regedit



Registry as seen by Thinstalled Office



Block based Streaming



Any network storage device can serve as a streaming server for hundreds or thousands of client PCs.

To use VMware Project North Star (Thinstall) packages in a streaming fashion, simply place your VMware Project North Star (Thinstall) package in a location that is accessible to client PCs and send a link to end-users to run the application directly, for example `\\server\sharename\application.exe`. You can also create shortcuts on the end-user's desktop which point to the centrally hosted EXE package(s). When the user clicks on the shortcut, the application will automatically begin streaming to the client PC. During the initial streaming startup process, the VMware Project North Star (Thinstall) statusbar will inform the user of the progress. Once enough of the application has been streamed that the application can begin running, the status bar will slide down and the user can begin using the application. As more parts of the application are required by the application, they will be pulled from the "streaming server". VMware Project North Star (Thinstall) does not require any special server software to provide streaming capability; any windows file share, NAT device, or SMB share can automatically provide this capability. The amount of data that needs to be transferred before the application can begin running will vary widely; Office 2003 requires only a fraction of the package contents to be streamed before the application can run. VMware Project North Star's streaming is currently designed for LAN based environments with 100mb networks recommended. For WAN and internet deployments where unexpected disconnects are more common, we recommend either deploying using a URL or using a desktop deployment solution to push the package in the background and only allow execution once the entire package has been downloaded. This will reduce failures and eliminate scenarios where unstreamed portions are required by the application during a network outage. A company with many branch offices would typically designate one "application repository" at each branch office which mirrors a central shared folder to optimize local performance for clients located at that branch.

For best security, it is recommended that a central shared directory be made read-only so users can read the package contents but they cannot tamper with the executable contents. As multiple users stream a package from a shared location, changes made by the application will be stored in the user's sandbox. The user's sandbox location defaults to `%AppData%\Thinstall\APPLICATION_NAME`, but can be configured at runtime or package time (Chapter 5, "Sandbox Overview," on page 65).A common setup is to have the user's sandbox be located on another central storage device so the user can walk up to any PC and retain their individual application settings centrally. When packages are being streamed from a central share, they will remain locked until all users have exited the application. VMware Project North Star (Thinstall) provides a mechanism for updating applications while they are still running to address this scenario ("Upgrading applications" on page 57).

Beta

Some common questions about VMware Project North Star's streaming:

Q. If I have a large package, will it take a long time to load over a network?

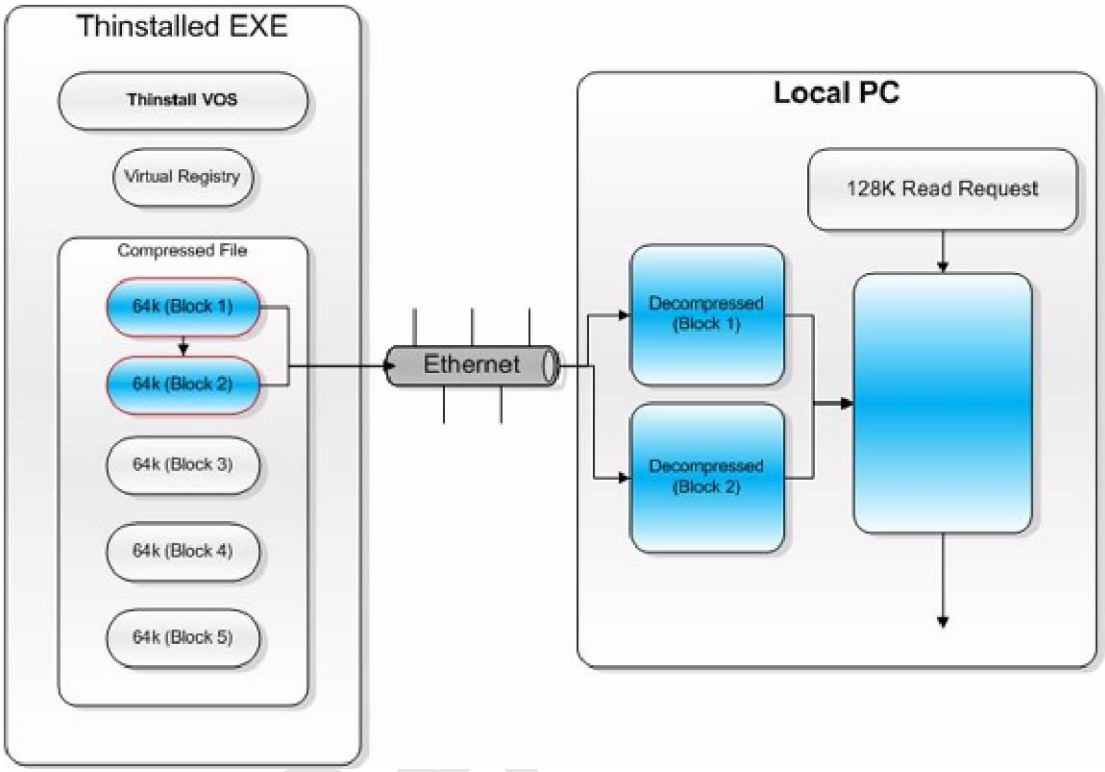
A. No, not necessarily. The size of the package has no effect on the startup time of an application. If you add an extra 20GB file to your package which is not used at runtime it will load at the same speed as before. If the application opens and reads 32k of data from the 20GB file, only 32kb of data is sent across the wire.

Q. There is no client and no server, how is streaming possible?

A. The "client" is built into the EXE package, but it is very small (400k) so the load time of the client across a network is only a few milliseconds. Once the client is loaded into memory on the client PC, it decides which blocks of data are further required from the "server" and reads them based on application activity.

When compressed VMware Project North Star (Thinstall) EXEs are placed on a network share or USB flash, the contents from the `EXE` file will automatically stream to client PCs in a block based fashion. As the application request specific parts of data files, this information is read in the compressed format over the network using the standard windows file-sharing protocol. After the data has been received on the client PC, data is decompressed directly into memory and provided to the application. Because no data needs to written to disk during the entire process the process is very fast and client PCs need only minimal disk space to host

the operating system. When subsequent read requests are made by the application for the same data, the Window disk cache will provide the same data without requiring a network read operation. If the client PC runs low on memory, Windows will automatically throw away some of its disk cache and provide the memory resource to other applications.



Supported OSs & Apps

OS Supported

- 32bit platforms: Windows NT, 2000, 2000 Server, XP, XPE, 2003 Server, Vista
- 64bit platforms: Windows XP 64 bit, Windows 2003 64 bit, Windows Vista 64 bit
- VMware Project North Star (Thinstall) does not support 16bit or non-intel platforms such as Windows CE

VMware Project North Star (Thinstall) currently supports most 16 and 32 bit windows applications running on 32bit and 64bit versions of Windows

- VMware Project North Star (Thinstall) can run 16/32bit applications on a 64bit OS, but it does not currently support 64 bit native applications
- Neither Windows nor VMware Project North Star (Thinstall) support 16bit applications running on 64 bit OSs

Software which typically needs to be deployed using traditional installation technologies:

- Applications requiring installation of kernel-mode device drivers (ODBC drivers will work because they are user-mode)
- Products such as anti-virus and personal firewalls
- Scanner drivers and printer Drivers
- Some VPN clients

The VMware Project North Star (Thinstall) Virtual OS (packaged with each EXE) is designed to run with no dependencies from the local system and can run directly from a network share or removable storage device in restricted user accounts with no changes to the local registry or filesystem. Because the Virtual OS is packaged directly into built applications there is no prior installation required and users can execute Thinstalled applications directly on clean or dirty installs of Windows.

In some cases VMware Project North Star (Thinstall) provides backwards compatibility and can allow applications to run on older operating systems, however in general the original application will retain it's minimum system requirements when running under VMware Project North Star's Virtual OS. For example, if the original application does not natively support Windows NT because it relies on Windows XP specific features, it will probably not execute under Windows NT using VMware Project North Star (Thinstall).

Desktop Integration

Thinstalled applications can interact with other applications installed on the desktop. The various ways applications typically interact the other components on the desktop are discussed below.

Cut & Paste

1 Pasting from System installed app to Thinstalled Application. This scenario has no limitations, the virtual application can receive any standard clipboard formats (text, graphics, html, etc.), as well the virtual application can receive OLE objects.

2 Pasting from Thinstalled application to System Application. In this scenario, the system application can receive any standard clipboard format, and OLE objects will automatically be converted to standard the closest clipboard.

Printers

Thinstalled applications have full normal access to any printer installed on the PC where it is running. There is no difference in printing ability for Thinstalled applications and system installed applications. Printer drivers themselves cannot be virtualized using VMware Project North Star (Thinstall) and must be installed normally on the PC.

Drivers

Thinstalled applications have full normal access to any device driver installed on the PC where it is running. There is no difference in access to installed device drivers for Thinstalled applications and system installed applications. If an application requires a device driver, this must be installed separately from the VMware Project North Star (Thinstall) package. In some cases the application may function correctly but in a limited fashion without an associated driver, for example Adobe Acrobat installs a printer driver which allows applications system-wide to render PDF files through their print mechanism. In that scenario, Adobe Acrobat (the application) can be used to load, edit, and save PDF files without installation, but other applications will obviously not see a new printer driver unless it is actually installed.

Access to local disk, removable disk, network shares

When creating a new project structure, VMware Project North Star (Thinstall) will automatically configure isolation modes for directories and registry subtrees. The isolation modes control which directories the application can read and write on the local PC. Though the configuration can be easily modified, the default options work well. The default options are:

- Fixed disk (i.e. `c:\`): The user can write to their Desktop and My Documents folder, other modifications made by the application will go into the user+app sandbox (located by default under Application Data). It is a simple configuration change to allow the user/application to write to any location on the PC.
- Removable Disk: By default, the user can read or write to any location on a removable disk assuming they have access rights to do so.
- Network mapped drives:By default, the user can read or write to any location on a network mapped disk assuming they have access rights to do so.

Beta

- UNC Network Paths: The user can read or write to any location on a removable disk assuming they have access rights to do so.

Access to the system registry

By default, Thinstalled application can read the full system registry as permitted by access permissions except for specific parts of the registry are automatically configured to be isolated from the system during the package creation process to reduce conflicts between different versions of virtual applications and system installed applications. By default, all registry writes from Thinstalled applications will be saved in an isolated sandbox and the system remains unchanged. This can be configured

Networking & Sockets

Thinstalled applications have full normal access to networking functionality, they can bind to local ports and make remote connections (assuming the user has access permissions to do these operations normally).

Shared memory Named Pipes

Thinstalled applications can interact with other applications on the system using shared memory, named pipes, mutexs, and semaphores.

VMware Project North Star (Thinstall) has the ability to isolate shared memory objects and synchronization objects so they are not visible to other applications, and other applications objects are not visible to your Thinstalled application.

COM, DCOM, and Out of process COM

Thinstalled applications can create COM controls both from the virtual environment and the system. If the system has a COM control installed as out-of-process COM, these controls will be executed as virtual processes when used by a Thinstalled application, enabling you to control modifications made by these applications. COM objects provided by virtualized applications will not be visible to other applications on the system unless the application is running in the same virtual environment. It is possible to execute system applications inside of virtual environments so they can access COM objects provided by virtual environments, for example a system installed application can access virtualized Office components by running the system application in the virtualized environment.

Services

Thinstalled applications can start and execute system installed services as well as virtual services. By default system services will be executed in the virtual environment so that modifications made by that service will be controlled by the virtual environment.

Filetype Associations

Thinstalled applications can execute system installed application via file type association. File type association can be added to the local PCs registry to point to Thinstalled EXE files on a per-user or per-machine basis.

Upgrading applications

VMware Project North Star (Thinstall) allows you to upgrade or roll-back an application version which it is still running. The upgrade process will occur automatically when the user quits the application and runs it a second time. In terminal server environments, you can have multiple users executing different versions at the same time during the transition period. The process for doing in-place upgrades or roll-backs is very simple and is described below.

Locked files

While applications are running, the EXE package will be locked and it cannot be replaced, deleted, or moved. This file lock ensures that any computer or user accessing a specific version of an application will continue to have that version available as long as their application process and subprocesses are running.

If an application is hosted in a central location and run by many users, this file lock means you cannot replace a packaged EXE with a new version until all users have released their locks on the file. This is not always easy to accomplish without using VMware Project North Star (Thinstall).

Beta

Upgrade Mechanism

VMware Project North Star (Thinstall) has an upgrade mechanism that allows you to deploy application upgrades even while older version are still in use by users. This mechanism is as simple as copying the new version into the old deployment directory with a .1, .2, .3, ... filename extension.

Example:

1 Deploy Version 1.0 of `myapp.exe`

   Copy application to central share at `\\server\share\myapp.exe` (`c:\program files\myapp\myapp.exe` is also another typical location)

   Push a desktop / start menu shortcut to user's desktop which points to shared EXE location at `\\server\share\myapp.exe`

2 After some time 2 users (user1 and user2) have launched `myapp.exe` and continue to use it

3 Now, you'd like to deploy Version 2.0 of `myapp.exe`, but you can't because the file is locked by 2 users. The solution: Copy Version 2.0 of `myapp.exe` to central share at `\\server\share\myapp.1`

After step 3, anytime a new user executes `\\server\share\myapp.exe` it will automatically relaunch using the package data in myapp.1

In this fashion, users shortcuts do not need to be updated; when you upgrade an application, the shortcuts continue to point to `myapp.exe`

If user1 and user2 quit the original version 1.0 of `myapp.exe` and reexecute it, they will start using version 2.0.

You can deploy version 3.0 of my app by placing it in the same directory with a higher number at the end.

Copy Version 2.0 of `myapp.exe` to central share at `\\server\share\myapp.2`

Once myapp.1 becomes unlocked you can delete it but `myapp.exe` should remain in place since the user shortcuts continue to point here. VMware Project North Star (Thinstall) will always use the filename which has the highest version number, so if you need to accomplish a roll-back to an earlier version and the most recent version is still locked, you need to copy the old version so it has the highest version number.

Sandbox considerations

When you release an upgrade to an application, you can control whether the user continues to use his or her old settings by keeping the sandbox name consistent in your `package.ini` file. By deploying a new application version with a new sandbox name, each user will create a separate sandbox and start with a fresh install maintaining isolated settings. If you deploy with the same sandbox name, users will keep their old settings.

NOTE Security Implications

Because apps can be upgraded by dropping in a new file in the same directory as the main application, the directory (and not just the file) where an application is hosted should be read-only for normal users.

Downloading upgrades

NOTE If you are a software developer and want to download new versions of your application from the Internet, you should store the temporary download files in a separate filename/directory until the download completes. Once the download has finished you can use an atomic MoveFile operation to move the file from the temporary location to the application's directory so that the next time the application restarts it will use the newer version. You must use an atomic operation because applications using MoveFile in the virtual environment may have sandboxing / WriteCopy applied.

Beta

Limitations

VMware Project North Star (Thinstall) does not support 100% of applications and in some cases application functionality may be degraded.

Before attempting a major application, it is often useful to check Thintalk (https://thinstall.com/account/index.php), our online discussion forum to find out what experience others have had with the same application. Most people only post to Thintalk when they have problems, so if you don't find any information there is a good chance the application is compatible with VMware Project North Star (Thinstall). VMware Project North Star (Thinstall) engineers address many compatibility issues with each new version, so it's a good idea to try out the latest version if you run into problems.

Some known limitations are listed below.

Device Drivers

Applications that require device drivers will not work when packaged with VMware Project North Star (Thinstall), unless those device drivers are natively installed on the host PC. For this reason Antivirus, VPN clients, personal firewalls, and disk/volume mounting-related utilities are usually not good candidates to be Thinstalled. In addition, applications that make their functionality available by inserting a fake or real printer driver will lose such functionality when Thinstalled.

Shell-integration

Some applications may have reduced functionality when Thinstalled because they are no longer integrated with other applications on the system. For example, applications that provide shell-integration will lose this functionality.

Network visible DCOM services

Because VMware Project North Star (Thinstall) isolates COM and DCOM from the outside world, applications that install network accessible DCOM services will only be accessible on the local PC by other Thinstalled applications running in the same sandbox. VMware Project North Star (Thinstall) supports virtual DCOM and COM on the same PC; it is only network DCOM that is affected.

Plugins and Addins

A different strategy for deploying and using Plugins and Addins ("Plugins and Addins" on page 154) should be considered since Thinstalled applications are isolated from other components on the PC.

Global Hook DLLs

Some applications use the API function SetWindowsHookEx to inject a DLL into all processes on the host PC. The injected DLL can intercept windows messages. This has been used to steal or spy on keyboard and mouse input from other applications. VMware Project North Star (Thinstall) will silently ignore requests made by applications trying to install Global Hook dlls via SetWindowsHookEx.

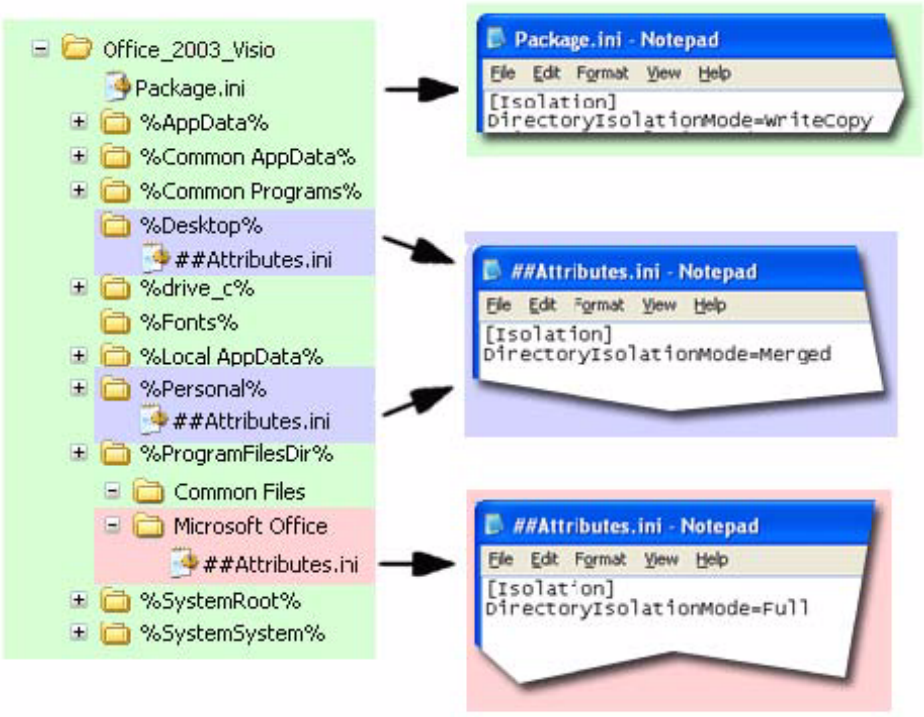
Beta

Beta

Isolation Modes

4

Isolation modes can be controlled on per-directory and per-registry subtree basis. By default, Setup Capture will produce projects that are configured for a locked-down desktop model as shown below. In a locked-down PC environment, the user will typically have write permission only to their Desktop, their `My Documents` folder, network shares, and removable disks. The default isolation mode for the project can be controlled in `package.ini`; this property will be inherited by child directories unless they explicitly override the isolation mode. For non-locked-down desktop models, setting the default isolation mode in `Project.ini` to "Merged" instead of "WriteCopy" allows the user to save files directly to the host filesystem at any location. The main risk for changing the default isolation mode to "Merged" is that there is potential for the application to leave behind residue when executing. For example, many shareware applications try to leave first-execution markers in random locations on the PC as part of their licensing system. For more details about understanding isolation mode settings, see "Understanding Isolation Modes" on page 62. Any disk writes to directories that are set with "WriteCopy" or "Full" isolation will be redirected to the sandbox. For more information about the location of the sandbox see Chapter 5, "Sandbox Overview," on page 65.



The following Isolation modes can be applied to subdirectories and registry subkeys.

WriteCopy

- System elements at this location will be visible to application If a system element and virtual element exist at the same location, the application will see the virtual element.
- Modifications to virtual elements go to sandbox
- Modifications to system elements go to sandbox
- New elements will be created in the sandbox

Merged

- System elements at this location will be visible to application If a system element and virtual element exist at the same location, the application will see the virtual element.
- Modifications to virtual elements go to sandbox
- Modifications to system elements go to system
- New elements will be created on the system

Full

- System elements at this location will not be visible to application
- Modifications to virtual elements go to sand box
- System elements cannot be read or modified
- New elements will be created in the sandbox

Q. How do I change the isolation mode for a directory?

A. Edit the `##Attibutes.ini` file in your package located in the directory where you want to modify the default isolation mode.

Q. How do I change the isolation mode for a registry subtree?

A. Edit the `HKEY_XXXXXX.txt` file. Each registry subtree will begin with the isolation mode.

NOTE VMware Project North Star (Thinstall) caches the isolation modes for the registry and filesystem at runtime in the sandbox. If you change the isolation mode for your project and then rebuild the `EXE` file, you may need to delete the sandbox in order for the change to take effect.

Understanding Isolation Modes

VMware Project North Star (Thinstall)'s Isolation Modes help solve the following classes of problems:

Problem 1: Application fails to run due to previous or future versions existing simultaneously or not uninstalling correctly

Solution: VMware Project North Star (Thinstall) "hides" host PC files/registry keys from the application when the host PC files are located in the same directories/subkeys created by the application's installer. VMware Project North Star (Thinstall) calls this Full Isolation (Chapter 4, "Isolation Modes," on page 61); for directories and subkeys that have Full Isolation, the application will not be aware of any of the host PC's files that might exist, and the application sees only virtual files and subkeys at fully isolated locations.

Problem 2: Application fails because it was not designed or tested for multi-user environments and expects it can modify files and keys without impacting other users

Problem 3: Application fails because it expects write permission to global locations and was not designed for locked-down desktop environments found in corporate environments or Windows Vista.

Beta

Solution: VMware Project North Star (Thinstall) makes copies of registry keys and files written by the application and performs all the modification in a user-specific sandbox. VMware Project North Star (Thinstall) calls this WriteCopy Isolation (Chapter 4, "Isolation Modes," on page 61); for directories and subkeys that have WriteCopy isolation, the application can see both the host PC's files and virtual files; however, all write operations will convert host PC files into virtual files in the sandbox.

VMware Project North Star (Thinstall) has 3 different isolation modes (Chapter 4, "Isolation Modes," on page 61), which are automatically determined by SetupCapture. SetupCapture has a few simple rules for determining what isolation mode to apply to a registry subtree or directory during capture.

- If the application created a new directory or registry subtree during its installation (on a clean PC), the isolation mode is set to Full Isolation
- User-specific storage areas like the Desktop and My Documents are set to Merged Isolation so the application has direct write access to these locations
- All other directories and subkeys will default to WriteCopy Isolation

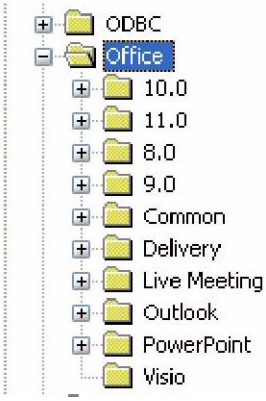
---

NOTE Network shares are not affected by isolation modes; read and write operations to network shares occur unchanged by VMware Project North Star (Thinstall).

For example, the following image shows a section of the Windows registry for a PC which has various older Office applications installed. Office 2003 creates the registry subtree:

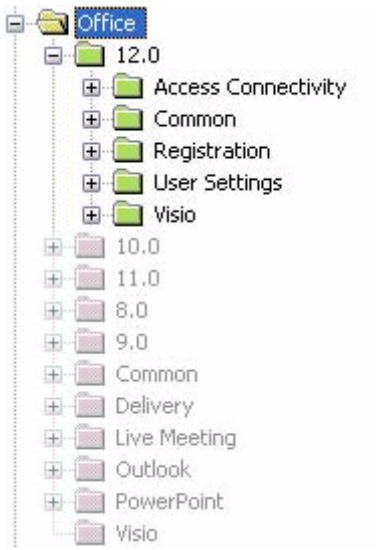
`HKEY_LOCAL_MACHINE\Software\Microsoft\Office\11.0`

Registry as seen by Windows Regedit



When running a thinstalled version of Visio 2007, VMware Project North Star (Thinstall) will set the `HKLM\Software\Microsoft\Office` registry subtree to Full Isolation. This setting prevents Visio 2007 from failing because of registry settings that may pre-exist on the host PC at the same location.

Registry as seen by Thinstalled Visio 2007



Beta

Sandbox Overview 5

The sandbox holds runtime modifications that applications make as they are running. The original EXE that you have built never changes, so it can be placed in a shared folder with read-only access.

The name of a sandbox (SANDBOXNAME) is specified in the `package.ini` file prior to build (Chapter 8, “Package.ini format,” on page 95).

Under normal circumstances, the sandbox is stored under `%AppData%\Thinstall\SANDBOXNAME`. On Windows XP this path would be `C:\Documents and Settings\USERNAME\Application Data\SANDBOXNAME`.

During startup, VMware Project North Star (Thinstall) searches for an existing Sandbox in the following locations in the order listed below. If a sandbox already exists, VMware Project North Star (Thinstall) will use the existing sandbox rather than creating a new one.

Search using an application-specific environment variable

1 %SANDBOXNAME_SANDBOX_DIR%.ComputerName

Example: If SANDBOXNAME=“Mozilla Firefox 1.0" then the environment variable “Mozilla Firefox 1.0_SANDBOX_DIR.ComputerName” is consulted

2 %SBNAME_SANDBOX_DIR%

Search using a global environment variable that applies to all Thinstalled applications

3 %THINSTALL_SANDBOX_DIR%.ComputerName\SANDBOXNAME

Example: THINSTALL_SANDBOX_DIR.ComputerName

4 %THINSTALL_SANDBOX_DIR%\SANDBOXNAME

Search for an application-specific ‘local’ sandbox

5 ExeDirectory\SANDBOXNAME.ComputerName

6 ExeDirectory\SANDBOXNAME

Search for a ‘local’ sandbox for all Thinstalled applications

7 ExeDirectory\Thinstall\SANDBOXNAME.ComputerName

8 ExeDirectory\Thinstall\SANDBOXNAME

Use the value “SandboxPath=” from Package.ini to determine a custom sandbox path this option is set (“SandboxPath” on page 112)

9 SANDBOXPATH\SANDBOXNAME.ComputerName

10 SANDBOXPATH\SANDBOXNAME

Beta

Search in the user's "Application Data\Thinstall" folder for a sandbox

11 %AppData%\Thinstall\SANDBOXNAME.ComputerName

12 %AppData%\Thinstall\SANDBOXNAME

If an existing sandbox cannot be found, Thinstall will create a new sandbox. VMware Project North Star (Thinstall) creates a new sandbox using the following logic:

1 If SANDBOXNAME_SANDBOX_DIR environment variable is set, try to create a sandbox at this location

2 If THINSTALL_SANDBOX_DIR environment variable is set, try to create a sandbox at this location

3 If SANDBOXPATH `Package.ini` option is set, try to create a sandbox at this location

4 Try to create a sandbox in the user `%AppData%\Thinstall` folder

NOTE SANDBOXNAME and SANDBOXPATH come from the `Package.ini` file

Example Scenarios:

Scenario: I want to run from USB Flash and have my sandbox load and save to the same directory where the EXE is located

How to accomplish: Set "SandboxPath=." in your `package.ini` file in the [BuildOptions] section. ("SandboxPath" on page 112)

Scenario: My users have a mapped drive to store their files and I want the sandbox stored on the mapped drive.

How to accomplish: Option 1. Use "SandboxPath=z:\Sandbox" if the drive is always mapped to the same letter ("SandboxPath" on page 112)

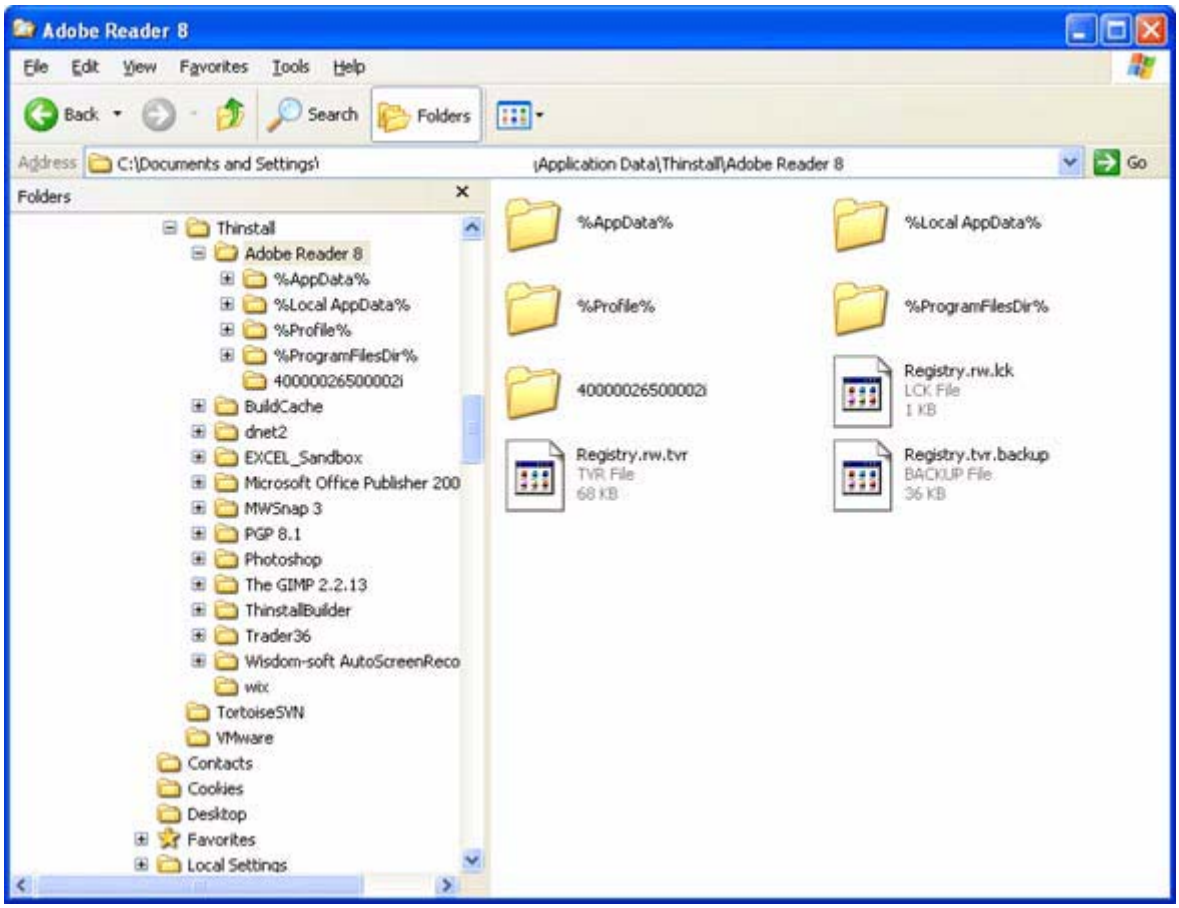
Option 2. In the user login script, set the environment variable "THINSTALL_SANDBOX_DIR=z:\Sandbox". Any Thinstalled application run after setting this variable will use `z:\Sandbox` for it's sandbox storage location.

Beta

Sandbox Structure

This information is provided for understanding purposes; we do not recommend modifications to the sandbox directory structure directly.

The sandbox is stored using a file structure nearly identical to the build project structure. Macro names ("Folder Macros" on page 148) are used for shell folder locations instead of hard-coded paths. This allows the sandbox to migrate to different PCs dynamically when the application is run from new locations.



In addition to shell folders like %AppData%, VMware Project North Star (Thinstall) will also create three registry files:

- `Registry.rw.tvr` (contains all registry modifications made by the application)
- `Registry.rw.lck` (zero byte lock file used to prevent other computers from simultaneously using a registry located on a network share)
- `Registry.tvr.backup` (contains a backup of the `.tvr` file from the last good execution, if corruption is detected on startup it will be restored)

Additionally, if an application executes a child process, VMware Project North Star (Thinstall) will create a directory who's name looks like the following:

```
40000026500002i\
```

This directory stores a dynamically generated EXE stub for the child process so the correct name is displayed in explorer for that child process. You can delete these directories and they will be regenerated the next time they are required.

Sandbox FAQ

Q. If I copy files into the sandbox folder, they are not visible to the application. Why not? VMware Project North Star (Thinstall) stores all of its filesystem information in the virtual registry; this allows it to optimize file system access in the virtual environment. For example, when an application tries to open a file, VMware Project North Star (Thinstall) does not need to consult the real filesystem twice to determine if the file exists or not (once for the real system location, and once for the sandbox location). Instead, VMware Project North Star (Thinstall) is able to check for the file's existence by only consulting the virtual registry, and, as a result, VMware Project North Star (Thinstall) can achieve great runtime performance. One artifact of this optimization is that if you copy files into the sandbox directory structure directly, the files will not be visible to the application. If the file already exists in the sandbox you can overwrite and update the file; however, we recommend leaving the sandbox alone and performing all modifications in the virtual environment instead.

Q. How can I view the virtual registry in the sandbox? The command-line utility "vregtool" can be used to list the contents of a virtual registry file as shown below. You may notice that some of the virtual file system data is stored in under the registry tree HKLM\FS.

vregtool registry.rw.tvr printkeys

```
sb_only HKEY_LOCAL_MACHINE\FS\%Profile%\.gimp-2.2\unitrc
deleted HKEY_LOCAL_MACHINE\FS\%Profile%\.fonts.cache-1.NEW
writecopy HKEY_LOCAL_MACHINE\Software writecopy HKEY_LOCAL_MACHINE\Software\Microsoft
writecopy HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
writecopy HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion
writecopy HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer
writecopy HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
writecopy HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell
              Folders
writecopy HKEY_LOCAL_MACHINE\Software\Thinstall
full   HKEY_LOCAL_MACHINE\Software\Thinstall\ProcessList
full   HKEY_LOCAL_MACHINE\Software\Thinstall\RuntimeObjects
full   HKEY_LOCAL_MACHINE\Software\Thinstall\SxS
writecopy HKEY_CURRENT_USER\Software writecopy HKEY_CURRENT_USER\Software\Microsoft
writecopy HKEY_CURRENT_USER\Software\Microsoft\Windows
writecopy HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
writecopy HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer
writecopy HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
writecopy
              HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\
              {c1664 b80-e468-11db-8906-806d6172696f}
writecopy
              HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\
              {c1664 b81-e468-11db-8906-806d6172696f}
writecopy
              HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\
              {c1664 b82-e468-11db-8906-806d6172696f}
writecopy
              HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\
              {c1664 b83-e468-11db-8906-806d6172696f}
writecopy HKEY_CURRENT_USER\Software\NVIDIA Corporation
writecopy HKEY_CURRENT_USER\Software\NVIDIA Corporation\Global
writecopy HKEY_CURRENT_USER\Software\NVIDIA Corporation\Global\nView
```

Draft

Utilities and Automation Tools 6

VMware Project North Star (Thinstall) utilities are stored in the default install directory (i.e. `c:\Program files\Thinstall.VS`)

"Setup Capture" on page 69

The primary application for converting traditional installers into VMware Project North Star (Thinstall) projects. SetupCapture is a GUI front-end for `snapshot.exe`.

The following utilities can be used to troubleshoot issues and help diagnose problems:

- "Log Monitor" on page 74

  Generates log files for running Thinstalled apps.

- "sbmerge" on page 85

  Can be used to merge runtime changes stored in the sandbox back into an existing project

- "dll_dump" on page 88

  Lists all running Thinstalled applications as well as individual DLLs loaded by those applications.

The following tools are intended to be used by companies integrating VMware Project North Star (Thinstall) into their products; these tools are automatically invoked during the build process, so you typically do not need to use them.

- VFTool

  Virtual Filesystem Compiler

- "VREGTool" on page 89l

  Virtual Registry Compiler,

- "TLink" on page 91

  VMware Project North Star (Thinstall) Linker

Setup Capture

Setup Capture is used to automatically convert standard applications to VMware Project North Star (Thinstall) projects, which can then be built into Thinstalled EXEs.

Before you use Setup Capture, a few planning steps to consider:

1 You should capture applications using a clean install on Windows ("Using a Clean PC for Capturing" on page 41)

> **NOTE** It's okay to install VMware Project North Star (Thinstall) on your clean machine for capturing purposes, it will not affect the snapshots

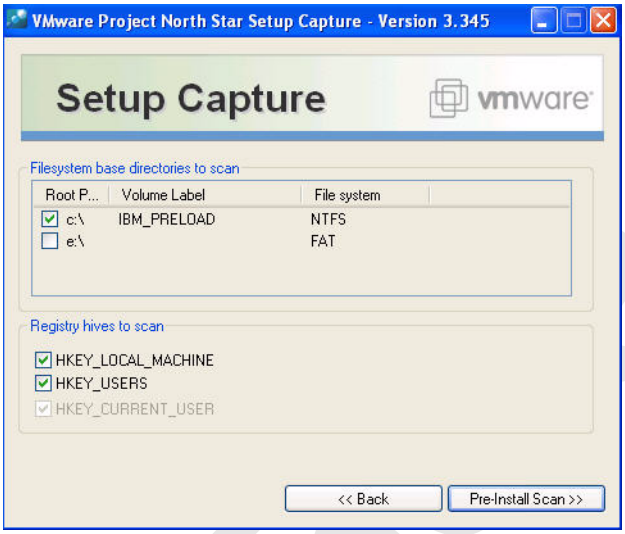
Beta

2 During installation, if given the option to install Shortcuts by the installer, select **Yes**. (Step 3, "Install the target application and configure it as you want the user see it.," on page 71)

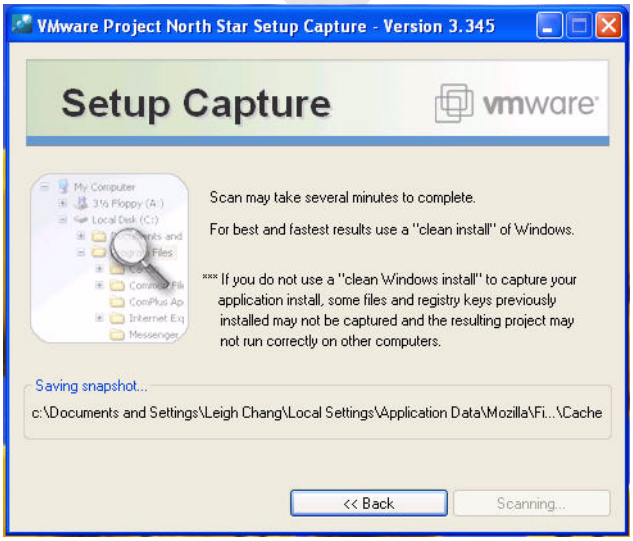
3 If your machine has background services running like anti-virus, it's a good a idea to stop them. If background applications create files or write registry values during the capture process, these changes will be included in the capture. Having extra data in a package generally does not cause a problem for applications, but it can increase the package size. Because Thinstalled applications are isolated from the system and VMware Project North Star (Thinstall) does not actually modify the registry and file system at runtime, *having extra data in a package will not harm a system as with MSI-based deployment systems*.

4 If the application requires a reboot during installation, Setup Capture will automatically continue the capture process ("Reboot continue" on page 73).

Step 1 Select which drives and registry hives to scan.

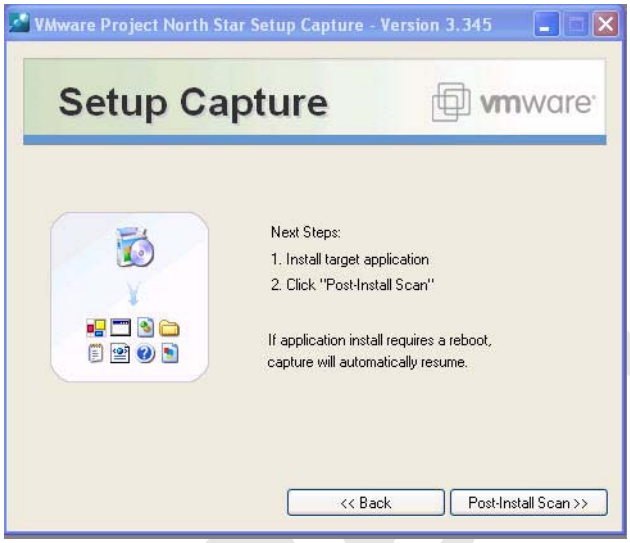


Step 2 Scan the machine to create an initial snapshot.

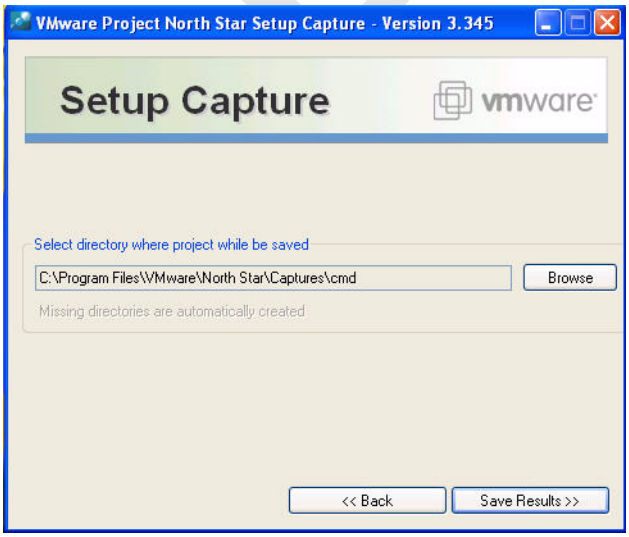

VMware Project North Star Setup Capture - Version 3.345
Setup Capture
vmware
Scan may take several minutes to complete.
For best and fastest results use a "clean install" of Windows.
*** If you do not use a "clean Windows install" to capture your application install, some files and registry keys previously installed may not be captured and the resulting project may not run correctly on other computers.
Saving snapshot...
c:\Documents and Settings\Leigh Chang\Local Settings\Application Data\Mozilla\Fi...\Cache
<< Back
Scanning...

Step 3 Install the target application and configure it as you want the user see it.

- If you execute the application after the installation completes, the capture will include per-user data settings (HKEY_CURRENT_USER, Documents & Settings). When a user executes the application, he or she will inherit these settings, but any additional changes made by the application at runtime will be saved to a user-specific sandbox.
- You can install required dependencies like .NET, ODBC connection drivers, Java, etc. These components will be included in the captured project. If these dependencies already existed on the system prior to starting the capture process, the package will not include them and they must be pre-installed on all machines on which the application will run.
- You can install application plugins at this point



Step 4 Do a final scan and select a directory to save project.



Step 5 Once the project has been saved, you can generate Thinstalled EXE files by running build.bat in the project directory.

NOTE If the application did not install any shortcuts, you may need edit `pacakge.ini` to tell VMware Project North Star (Thinstall) what EXEs you want to generate.("Package.ini generation" on page 72)

Package.ini generation

How Setup Capture decides which EXE will be included in a generated package.ini file

Most larger applications will install many different EXE files during installation; however, not all of the EXE files installed are meant to be used directly by users. During the Setup Capture process, VMware Project North Star (Thinstall) will look to see which Desktop and Start Menu shortcuts were created and use these links to generate VMware Project North Star (Thinstall) EXEs during build.

For example, Office 2003 installs 44 different EXEs:

However, Office 2003 only installs Start Menu entries for 13 of these 44 EXE files. Most of the EXE files are probably not intended to invoked directly by the user.

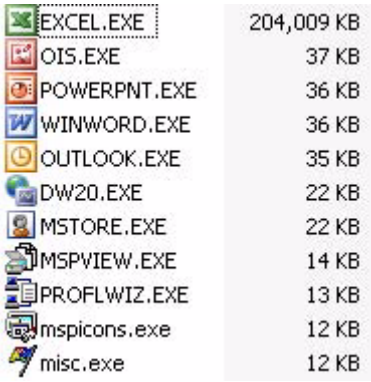
For a standard capture of Office 2003, VMware Project North Star (Thinstall) generates a `package.ini` which contains entries to generate EXEs for only the shortcuts installed by Office like this:

```
[EXCEL.EXE]
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\EXCEL.EXE
ReadOnlyData=bin\Package.ro.tvr

[OUTLOOK.EXE]
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\OUTLOOK.EXE
Shortcut=EXCEL.EXE

[POWERPNT.EXE]
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\POWERPNT.EXE
Shortcut=EXCEL.EXE

[WINWORD.EXE]
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\WINWORD.EXE
Shortcut=EXCEL.EXE
...
```

NOTE For each entry in the package file, there is a *Source=* option which specifies which application will be launched when building the package.

In this case, `[EXCEL.EXE]` is set up to host all of the package file/registry data by using *ReadOnlyData=*, while all the other `EXE` files simply reference the package data from `EXCEL.EXE` using *Shortcut=*

When building this packaging using build.bat, you'll see the following appear in your bin directory.



Because VMware Project North Star (Thinstall) uses shortcuts installed during the capture process to generate the package.ini, *make sure you let the installer create shortcuts for you*. Many installers offer the option to not write shortcuts; if you disable shortcut creation, Setup Capture won't know what EXEs you intend to generate and will start with an empty project.

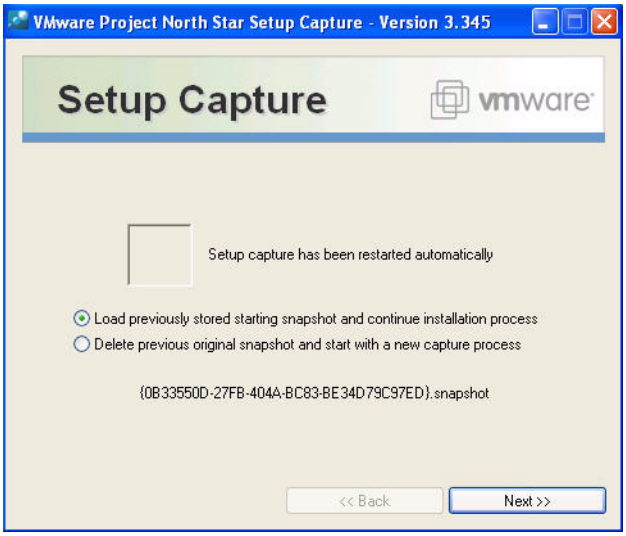
If you accidentally disabled shortcuts during install or the application does not install any shortcuts during it's installation process, you can manually tell VMware Project North Star (Thinstall) which EXEs the user can access directly by using a text editor to add this information to your package.ini file. Simply add the following lines anywhere in the text file.

```
[My Application.exe]
Source=%ProgramFilesDir%\My App\My Application.exe
ReadOnlyData=bin\Package.ro.tvr
```

Where %ProgramFilesDir%\My App\My Application.exe corresponds to a path inside of your VMware Project North Star (Thinstall) project.

Reboot continue

When capturing an application that requires a reboot during installation, just reboot the computer normally when prompted by the installer. Setup Capture should automatically restart and allow you to continue the capture process. If you ran Setup Capture from a network share instead of the host PC, it is possible the local PC will try to launch Setup Capture prior to the network becoming fully initialized, in this case you may need to manually run Setup Capture from the network share and it will prompt you to continue the capture process as shown below.


VMware Project North Star Setup Capture - Version 3.345
Setup Capture
vmware
Setup capture has been restarted automatically
Load previously stored starting snapshot and continue installation process
Delete previous original snapshot and start with a new capture process
{0B33550D-27FB-404A-BC83-BE34D79C97ED}.snapshot
<< Back
Next >>

Beta

Example:

1 Run Setup Capture

2 Start installation of Application XYZ, installer displays “must reboot to complete installation”

3 Reboot computer & Log in

4 Setup Capture should restart as shown above, click **Load previously stored starting snapshot...**

5 Finish any final installations steps

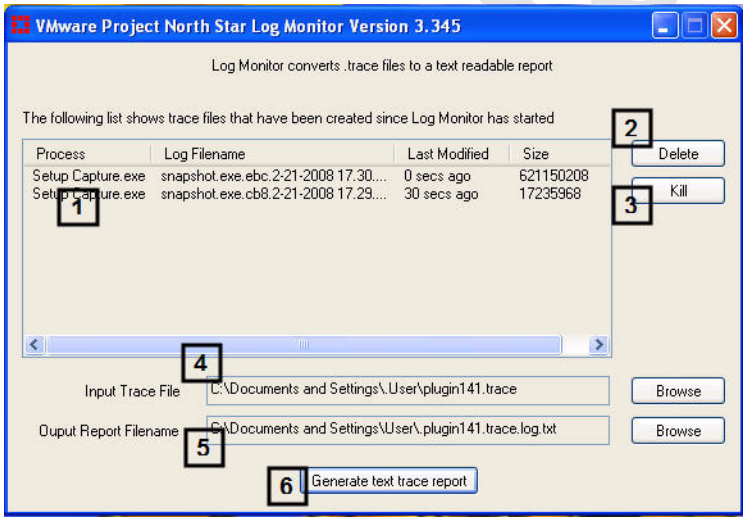
6 Click **Post Install Scan** on the Setup Capture dialog

Log Monitor

Log Monitor allows you to record detailed information about any application’s execution history for later review. A few things that are recorded:

- Win32 API calls made by applications running in the VMware Project North Star (Thinstall) VOS, along with parameter and result information
- A list of potential errors, exceptions, and security events within the application
- A list of all DLLs loaded by the application and address ranges

Log monitor can be found in your VMware Project North Star (Thinstall) install directory (typically `C:\Program Files\Thinstall.VS`) It can also be run from **Start Menu>Program Files>Thinstall Virtualization Suite>Log Monitor**



1 Any new VMware Project North Star (Thinstall) process which has been started after Log Monitor begins will show up in the list if the application was built with logging options enabled or with “Just-in-time” logging options enabled. Clicking on the process entry will automatically fill in #4 and #5.

2 Delete will delete any trace files selected in the process list (#1). This will not delete any text .log files you may have generated

3 Kill will kill any process selected if it is currently running. This is a easy way to stop a process from logging additional entries once an error condition has been reached.

4 Input trace file. You can manually browse for a trace file to convert by clicking the button to the right

5 Output report Files. This file will be generated when you click **Generate text trace report**. The file should be viewed with a text editor that supports unix-style line breaks such as Msdev, Wordpad, or Word (not notepad).

See: Tips and tricks for locating errors in trace files (“Locating Errors” on page 75)

Locating Errors

VMware Project North Star (Thinstall) logging functionality (created using Log Monitor)("Log Monitor" on page 74) provides a large amount of information. Here are a few tips you can follow to more easily zoom in on problems to investigate.

Tip 1. Use TextPad to view your log files, as it can handle fairly large files.

Most text file viewers cannot handle large log files, and VMware Project North Star (Thinstall) trace files can easily be larger than 100MB. If your trace file exceeds 200MB, you may need to chop off the beginning of the file using a utility like "tail.exe".

Tip 2. Look at "Potential Errors" section first.

The most important part of the log for diagnosing problems comes at the end of the log, in the section labeled "----Potential Errors Detected ---". You'll find this at the very bottom of your `.txt` trace file.

Many entries listed in the potential errors section are innocent and do not actually indicate an error. VMware Project North Star (Thinstall) list each Win32 API call where the windows error code changed, and most of these do not indicate a real problem which you need to focus on.

Red Flags:

- Exceptions

  Many applications generate exceptions while they are running; exceptions do not always indicate an error or fault, but are generally a good indicator. Exceptions have different types, (C++, .NET, Floating point, Access Violation, Data Execution Prevention, etc.). The trace file will record the type of exception generated and the DLL which generated the exception. If the application threw an exception from self-generating code, as is sometimes the case for .NET and Java applications, the trace will indicate "unknown_module".

  example trace entry for an exception:

  *** *Exception EXCEPTION_ACCESS_VIOLATION on read of 0x10 from unknown_module:0x7c9105f8*

  NOTE VB6 applications throw many floating point exceptions during their normal execution, you can ignore these. Once you find an exception, look higher in the trace file to see if you can locate the source of the exception.

- MSI error messages

  The MSI system will often kick in for applications that are not installed correctly and will try to perform self-repair. Unless a feature was installed using "on demand install", it usually indicates that there is something wrong in the environment. Luckily, MSI provides good indicators and error messages in the trace file before it starts self-repair. Search your trace file for calls to *FormatMessage*; this will also give you important information from the MSI installer.

Tip 3. The error may occur in a child process, make sure you look at the correct trace file.

Log monitor will produce one trace file for each process that is started. If your application starts several child processes, the first step is to determine which process is causing the problem. In some cases such as out-of-process COM, a parent application will use COM to launch a child process, execute a function remotely, and then continue executing.

Tip 4. When running applications from a network share, there is always 2 processes created—Ignore the first one.

In order to work-around Symantec AntiVirus slow performance issues, VMware Project North Star (Thinstall) always relaunches processes when running from a network share ("NetRelaunch" on page 108).

Beta

Tip 5. Search for the error message you see in dialog boxes.

Many applications will call the Win32 API function *MessageBox* to display unexpected errors at runtime. You can search your trace file for either "MessageBox" or the contents of the string displayed in the error dialog. For example, if the application displayed a cryptic error message "Unexpected child node / Press OK to continue," you could search for this string (or part of the string) and find what the application was doing just before the dialog box appeared.

Tip 6. Kill the application as soon as it gets to an error situation.

By killing the application as soon as possible, you will have less information to sift through in the trace file. If you kill the application immediately after it has an error, chances are good that you can look at the end of the trace and scan upwards to see the source of the problem. VMware Project North Star (Thinstall)'s Log Monitor utility has a handle "Kill" button to help you do this.

Tip 7. You can generate .txt versions of .trace files while an application is still running.

If needed, you can convert a `.trace` file into a text report while the application is still running.

Tip 8. Narrow your focus on calls originating from a specific DLL and thread.

VMware Project North Star (Thinstall)'s log format ("Log Format" on page 76) lets you see which DLL and thread is making a call. When tracking back from the source of an error, you can usually ignore other threads and often ignore calls from system DLLs.

Log Format

General API Log Message Format

The general format for API calls looks like this:

```
000257 0a88 mydll.dll :4ad0576d–>kernel32.dll:7c81b1f0 SetConsoleMode (IN HANDLE
hConsoleHandle=7h, IN DWORD dwMode=3h)
000258 0a88 mydll.dll :4ad0576d<–kernel32.dll:7c81b1f0 SetConsoleMode –>BOOL=1h ()
```

`000257`—Indicates the log entry number. Each log entry will have a unique number. The "----Potential Errors Detected ---" will refer back to individual log entries using its log entry number.

`0a88`—Indicates the currently executing thread ID. If the application has only one thread, this numberwill never change. Since log entries are recorded in order of execution, if two or more threads are recording data to the log file then you may need to use the thread ID to follow thread-specific sequential actions.

`mydll.dll`—Indicates the DLL which is making the API call

`4ad0576d`—Indicates the return address for the API call made by mydll.dll. Typically this will tell you the address in the code where the call is originating.

–> Indicates the call is being entered. For call entry log element, all of the input parameters are displayed (in and in/out parameters)

<–Indicates the call is returning to the original caller. For call exit log entries, all of the output parameters are displayed (out and in/out parameters).

`kernel32.dll`—Indicates the DLL where the API call is landing.

`7c81b1f0`—Indicates the address of the API inside of kernel32 where the call is landing. If you disassemble kernel32.dll at address 7c81b1f0, you will find the code for the function SetConsoleMode

`–>BOOL=1h` Indicates the API returned the value "1" and the return code has type "BOOL".

Application Startup Information

```
000001 0a88 Logging started for Module=C:\test\cmd_test\bin\cmd.exe
               Using archive=
               PID=0xec
               CommandLine = cmd
000002 0a88 Logging options: CAP_LEVEL=9 MAX_CAP_ARY=25 MAX_CAP_STR=150
MAX_NEST=100
```

Beta

```
VERSION=3.090

000003 0a88 System Current Directory = C:\test\cmd_test\bin Virtual Current Directory =
               C:\test\cmd_test\bin

000004 0a88 |start_env_var| =::=::\
000005 0a88 |start_env_var| =C:=C:\test\cmd_test\bin
000006 0a88 |start_env_var| =ExitCode=00000000
000007 0a88 |start_env_var| ALLUSERSPROFILE=C:\Documents and Settings\All Users.WINDOWS
...
...
...
```

List of DLLs loaded into memory during runtime

The section labeled "---Modules loaded ---" is located near the end of the log and will describe all of the DLLs which were loaded into memory at runtime and the addresses they were loaded to. This list also describes whether DLLs were loaded by Windows or by VMware Project North Star (Thinstall).

```
---Modules loaded --
               PRELOADED_MAP 00400000-00452fff, C:\Program Files\Adobe\Reader
8.0\Reader\AcroRd32.exe
               PRELOADED_BY_SYSTEM 00400000-00452fff, C:\Program Files\Adobe\Reader
8.0\Reader\AcroRd32.exe
               SYSTEM_LOADED 00400000-00452fff, C:\test\AcroRd32.exe
               SYSTEM_LOADED 00df0000-00df8fff, C:\WINDOWS\system32\Normaliz.dll
        MEMORY_MAPPED_ANON 013b0000-020affff, C:\Program Files\Adobe\Reader
8.0\Reader\AcroRd32.dll
               SYSTEM_LOADED 035a0000-035b4fff, C:\WINDOWS\system32\nvwddi.dll
               SYSTEM_LOADED 035a0000-03698fff, C:\WINDOWS\system32\nvwimg.dll
               SYSTEM_LOADED 035e0000-03ba9fff, C:\WINDOWS\system32\ieframe.dll
               SYSTEM_LOADED 04730000-04828fff, C:\WINDOWS\system32\nvwimg.dll
        MEMORY_MAPPED_ANON 05000000-050a8fff, C:\Program Files\Adobe\Reader
8.0\Reader\ACE.dll
        MEMORY_MAPPED_ANON 06000000-064b0fff, C:\Program Files\Adobe\Reader
8.0\Reader\AGM.dll
        MEMORY_MAPPED_ANON 07000000-07019fff, C:\Program Files\Adobe\Reader
8.0\Reader\BIB.dll
        MEMORY_MAPPED_ANON 08000000-08239fff, C:\Program Files\Adobe\Reader
8.0\Reader\CoolType.dll
               SYSTEM_LOADED 0ffd0000-0fff7fff, C:\WINDOWS\system32\rsaenh.dll
               SYSTEM_LOADED 10000000-10165fff, C:\WINDOWS\system32\nview.dll
               SYSTEM_LOADED 20000000-202c4fff, C:\WINDOWS\system32\xpsp2res.dll
         MEMORY_MAPPED_ANON 20800000-20fdafff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\AcroForm.api
        MEMORY_MAPPED_ANON 22100000-224f2fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Annots.api
        MEMORY_MAPPED_ANON 23000000-2311afff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\DigSig.api
        MEMORY_MAPPED_ANON 23800000-23951fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\EScript.api
        MEMORY_MAPPED_ANON 24000000-24023fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\EWH32.api
        MEMORY_MAPPED_ANON 25800000-25817fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\IA32.api
        MEMORY_MAPPED_ANON 26800000-2680ffff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\eBook.api
        MEMORY_MAPPED_ANON 28000000-285d1fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\PPKLite.api
        MEMORY_MAPPED_ANON 28800000-2885dfff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\reflow.api
        MEMORY_MAPPED_ANON 29000000-29227fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\MakeAccessible.api
        MEMORY_MAPPED_ANON 29800000-2985afff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Accessibility.api
        MEMORY_MAPPED_ANON 29a00000-29a1dfff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\ReadOutLoud.api
        MEMORY_MAPPED_ANON 2a000000-2a018fff, C:\Program Files\Adobe\Reader
```

Beta

```
8.0\Reader\plug_ins\Search5.api
        MEMORY_MAPPED_ANON 2a300000-2a359fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Search.api
        MEMORY_MAPPED_ANON 2a800000-2a821fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\SendMail.api
        MEMORY_MAPPED_ANON 2b000000-2b045fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Spelling.api
        MEMORY_MAPPED_ANON 2b800000-2b865fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\PDDom.api
        MEMORY_MAPPED_ANON 2d800000-2d94efff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Multimedia.api
        MEMORY_MAPPED_ANON 2e000000-2e02ffff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\weblink.api
        MEMORY_MAPPED_ANON 30800000-30829fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Updater.api
        MEMORY_MAPPED_ANON 31800000-31810fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\HLS.api
        MEMORY_MAPPED_ANON 32000000-3204dfff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\SaveAsRTF.api
        MEMORY_MAPPED_ANON 40800000-40821fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\DVA.api
        MEMORY_MAPPED_ANON 45800000-458cffff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\Checkers.api
        MEMORY_MAPPED_ANON 46800000-46873fff, C:\Program Files\Adobe\Reader
8.0\Reader\plug_ins\ImageViewer.API
                SYSTEM_LOADED 59a60000-59b00fff, C:\WINDOWS\system32\dbghelp.dll
                SYSTEM_LOADED 5ad70000-5ada7fff, C:\WINDOWS\system32\uxtheme.dll
                SYSTEM_LOADED 5d090000-5d129fff, C:\WINDOWS\system32\COMCTL32.dll
                SYSTEM_LOADED 61410000-61533fff, C:\WINDOWS\system32\urlmon.dll
                SYSTEM_LOADED 629c0000-629c8fff, C:\WINDOWS\system32\LPK.DLL
                SYSTEM_LOADED 63380000-633f7fff, C:\WINDOWS\system32\jscript.dll
                SYSTEM_LOADED 6e850000-6e894fff, C:\WINDOWS\system32\iertutil.dll
                SYSTEM_LOADED 71aa0000-71aa7fff, C:\WINDOWS\system32\WS2HELP.dll
                SYSTEM_LOADED 71ab0000-71ac6fff, C:\WINDOWS\system32\ws2_32.dll
                SYSTEM_LOADED 71bf0000-71c02fff, C:\WINDOWS\system32\SAMLIB.dll
                SYSTEM_LOADED 746c0000-746e8fff, C:\WINDOWS\system32\msls31.dll
                SYSTEM_LOADED 746f0000-74719fff, C:\WINDOWS\system32\msimtf.dll
                SYSTEM_LOADED 74720000-7476afff, C:\WINDOWS\system32\MSCTF.dll
                SYSTEM_LOADED 74d90000-74dfafff, C:\WINDOWS\system32\USP10.dll
                SYSTEM_LOADED 755c0000-755edfff, C:\WINDOWS\system32\msctfime.ime
                SYSTEM_LOADED 75cf0000-75d80fff, C:\WINDOWS\system32\MLANG.dll
                SYSTEM_LOADED 75e90000-75f3ffff, C:\WINDOWS\system32\SXS.DLL
                SYSTEM_LOADED 76390000-763acfff, C:\WINDOWS\system32\IMM32.DLL
                SYSTEM_LOADED 76780000-76788fff, C:\WINDOWS\system32\shfolder.dll
                SYSTEM_LOADED 769c0000-76a72fff, C:\WINDOWS\system32\USERENV.dll
                SYSTEM_LOADED 76b40000-76b6cfff, C:\WINDOWS\system32\WINMM.dll
                SYSTEM_LOADED 76bf0000-76bfafff, C:\WINDOWS\system32\PSAPI.DLL
                SYSTEM_LOADED 76f60000-76f8bfff, C:\WINDOWS\system32\WLDAP32.dll
                SYSTEM_LOADED 76fd0000-7704efff, C:\WINDOWS\system32\CLBCATQ.DLL
                SYSTEM_LOADED 77050000-77114fff, C:\WINDOWS\system32\COMRes.dll
                SYSTEM_LOADED 77120000-771abfff, C:\WINDOWS\system32\OLEAUT32.dll
                SYSTEM_LOADED 771b0000-7727efff, C:\WINDOWS\system32\WININET.dll
                SYSTEM_LOADED 773d0000-774d2fff,
C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2982_x-ww_ac3f9
                c03\comctl32.dll
                SYSTEM_LOADED 774e0000-7761cfff, C:\WINDOWS\system32\ole32.dll
                SYSTEM_LOADED 77690000-776b0fff, C:\WINDOWS\system32\NTMARTA.DLL
                SYSTEM_LOADED 77920000-77a12fff, C:\WINDOWS\system32\SETUPAPI.dll
                SYSTEM_LOADED 77b40000-77b61fff, C:\WINDOWS\system32\appHelp.dll
                SYSTEM_LOADED 77c00000-77c07fff, C:\WINDOWS\system32\VERSION.dll
                SYSTEM_LOADED 77c10000-77c67fff, C:\WINDOWS\system32\msvcrt.dll
                SYSTEM_LOADED 77dd0000-77e6afff, C:\WINDOWS\system32\ADVAPI32.dll
                SYSTEM_LOADED 77e70000-77f00fff, C:\WINDOWS\system32\RPCRT4.dll
                SYSTEM_LOADED 77f10000-77f56fff, C:\WINDOWS\system32\GDI32.dll
                SYSTEM_LOADED 77f60000-77fd5fff, C:\WINDOWS\system32\SHLWAPI.dll
                SYSTEM_LOADED 77fe0000-77ff0fff, C:\WINDOWS\system32\Secur32.dll
    MEMORY_MAPPED_ANON 78130000-781cafff,
```

```
C:\WINDOWS\WinSxS\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.163_x-
                ww_681e29fb\msvcr80.dll
    MEMORY_MAPPED_ANON 7c420000-7c4a6fff,
C:\WINDOWS\WinSxS\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.163_x-
                ww_681e29fb\msvcp80.dll
    MEMORY_MAPPED_ANON 7c4c0000-7c53cfff,
C:\WINDOWS\WinSxS\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.163_x-ww_681e29fb\msvcm80.dll
                SYSTEM_LOADED 7c800000-7c8f3fff, C:\WINDOWS\system32\kernel32.dll
                SYSTEM_LOADED 7c900000-7c9affff, C:\WINDOWS\system32\ntdll.dll
                SYSTEM_LOADED 7c9c0000-7d1d4fff, C:\WINDOWS\system32\SHELL32.dll
                SYSTEM_LOADED 7e410000-7e49ffff, C:\WINDOWS\system32\USER32.dll
                SYSTEM_LOADED 7e830000-7eb9efff, C:\WINDOWS\system32\mshtml.dll
```

SYSTEM_LOADED—Indicates the DLL was loaded by Windows, the file must exist on the disk.

MEMORY_MAPPED_ANON—Indicates the DLL was loaded by VMware Project North Star (Thinstall), the file may be loaded from the virtual file system.

46800000-46873fff—Indicates the address ranges in virtual memory where the DLL was loaded.

```
----Timing Report: list of slowest 150 objects profiled ---

8255572220 total cycles (2955.56 ms): |sprof| thinstall_LoadLibrary2

765380728 cycles (274.01 ms) on log entry 21753
428701805 cycles (153.48 ms) on log entry 191955
410404281 cycles (146.93 ms) on log entry 193969
231503734 cycles (82.88 ms) on log entry 188438
227419794 cycles (81.42 ms) on log entry 190209
211952538 cycles (75.88 ms) on log entry 197416
202095103 cycles (72.35 ms) on log entry 189394
200356604 cycles (71.73 ms) on log entry 194646
192420627 cycles (68.89 ms) on log entry 190812
183214731 cycles (65.59 ms) on log entry 195836
... 438 total calls
7847975891 total cycles (2809.64 ms): |sprof| ts_load_internal_module
764794646 cycles (273.80 ms) on log entry 21753
426837866 cycles (152.81 ms) on log entry 191955
408570540 cycles (146.27 ms) on log entry 193969
228790905 cycles (81.91 ms) on log entry 188438
224240114 cycles (80.28 ms) on log entry 190209
209789307 cycles (75.11 ms) on log entry 197416
200287437 cycles (71.70 ms) on log entry 189394
198429210 cycles (71.04 ms) on log entry 194646
190612618 cycles (68.24 ms) on log entry 190812
180322247 cycles (64.56 ms) on log entry 195836
... 94 total calls
4451728477 total cycles (1593.76 ms): |sprof| ts_lookup_imports
544327945 cycles (194.87 ms) on log entry 21758
385149968 cycles (137.89 ms) on log entry 193970
187246661 cycles (67.04 ms) on log entry 190210
173617241 cycles (62.16 ms) on log entry 194647
173481875 cycles (62.11 ms) on log entry 19065
148587579 cycles (53.20 ms) on log entry 195837
133165053 cycles (47.67 ms) on log entry 189395
126806624 cycles (45.40 ms) on log entry 197417
125894370 cycles (45.07 ms) on log entry 200296
121213253 cycles (43.40 ms) on log entry 200657
... 34 total calls
1099873523 total cycles (393.76 ms): |sprof| new_thread_start
561664565 cycles (201.08 ms) on log entry 151922
531551734 cycles (190.30 ms) on log entry 152733
1619002 cycles (0.58 ms) on log entry 72875
1554448 cycles (0.56 ms) on log entry 637896
1481627 cycles (0.53 ms) on log entry 72881
1091972 cycles (0.39 ms) on log entry 580771
```

Beta

Potential Errors

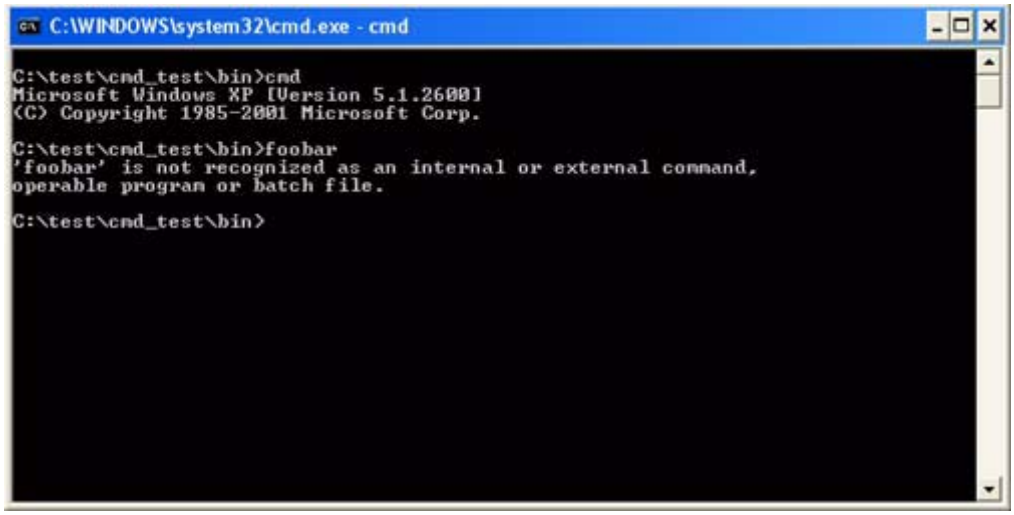
The potential errors is a collection of all log entries which have "***" in their string. VMware Project North Star (Thinstall) marks entries that could potentially be a problem by adding "***" to the log entry output. See Locating Errors for more tips on interpreting this section.

```
----Potential Errors Detected ---

006425 0000075c        LoadLibraryExW 'C:\Program Files\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
006427 0000075c        LoadLibraryExW 'C:\Program Files\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL' flags=2
-> 0 (failed ***)
006428 0000089c nview.dll :1005b94b<-kernel32.dll:7c80ae4b *** LoadLibraryW -
>HMODULE=7c800000h () *** GetLastError() returns 2 [0]: The system cannot find the file
specified.
007062 0000075c        LoadLibraryExW 'C:\Program Files\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
010649 0000075c        LoadLibraryExW 'C:\Program Files\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL'
flags=2 -> 0 (failed ***)
019127 0000075c MSVCR80.dll :781348cc<-msvcrt.dll :77c10396 *** GetEnvironmentVariableA -
>DWORD=0h (OUT LPSTR lpBuffer=*0h <bad ptr>) *** GetLastError() returns 203 [0]: The system
could not find the environment option that was entered.
019133 0000075c MSVCR80.dll :78133003<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [203]: The specified procedure could not be found.
019435 0000075c MSVCR80.dll :78136e08<-dbghelp.dll :59a60360 *** GetFileType ->DWORD=0h ()
*** GetLastError() returns 6 [0]: The handle is invalid.
019500 0000075c MSVCR80.dll :78134481<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [0]: The specified procedure could not be found.
019530 0000075c MSVCR80.dll :78131dcd<-dbghelp.dll :59a603a1 *** GetModuleHandleA -
>HMODULE=0h () *** GetLastError() returns 126 [0]: The specified module could not be found.
```

Forensics Example

To illustrate how Log Monitor ("Log Monitor" on page 74) can be used to delve into application problems, in this section uses an example to illustrate. Here `cmd.exe` is packaged with VMware Project North Star (Thinstall) and run with logging being recorded.

To simulate an application behaving incorrectly, a simple invalid command is issued. In this case, we have requested `cmd.exe` to execute the command `foobar`, and `cmd.exe` prints out the message "foobar is not recognized as an internal or external command." By viewing the resulting trace file we can dig into what `cmd.exe` is doing in much greater detail and learn how it operates. All applications will manifest misbehavior in different ways, so there is no one set way to track down issues.



The first place to check in a log file is the section near the end labeled "----Potential Errors Detected ---".

Beta

In this section, you can find all the API functions in which the GetLastError code was modified. The paths highlighted in bold indicate locations that `cmd.exe` was looking for foobar, and paths in red indicate locations in the virtual file system that were probed for these file system probes.

```
----Potential Errors Detected ---
*** Unable to determine if any services need to be auto-started, error 2
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW ->HANDLE=ffffffffh .. *** GetLastError() returns 2 [203]: The system cannot
find the file specified.
*** FindFirstFileW 'C:\test\cmd_test\bin\foobar' -> INVALID_HANDLE_VALUE *** failed [FS
missing in view 0][fs entry not found %drive_C%\test\cmd_test\bin\foobar][fs entry not found
%drive_C%\test\cmd_test\bin]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system
probe C:\WINDOWS\system32\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing
in view 0][fs entry not found %SystemSystem%\foobar]
*** FindFirstFileW 'C:\WINDOWS\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system probe
C:\WINDOWS\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing in view
0][fs entry not found %SystemRoot%\foobar]
*** FindFirstFileW 'C:\WINDOWS\System32\Wbem\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe C:\WINDOWS\System32\Wbem\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\System32\Wbem\foobar' -> INVALID_HANDLE_VALUE *** failed [FS
missing in view 0][fs entry not found %SystemSystem%\Wbem\foobar]
*** FindFirstFileW 'c:\program files\subversion\bin\foobar.*' -> INVALID_HANDLE_VALUE ***
failed [system probe c:\program files\subversion\bin\foobar.* -> ffffffffh][no virtual or
system matches]
*** FindFirstFileW 'c:\program files\subversion\bin\foobar' -> INVALID_HANDLE_VALUE *** failed
[FS missing in view 0][fs entry not found %ProgramFilesDir%\subversion\bin\foobar][fs entry
not found %ProgramFilesDir%\subversion\bin]
*** FindFirstFileW 'c:\Program Files\Microsoft SQL Server\90\Tools\binn\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe c:\Program Files\Microsoft SQL
Server\90\Tools\binn\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'c:\Program Files\Microsoft SQL Server\90\Tools\binn\foobar' ->
INVALID_HANDLE_VALUE *** failed [FS missing in view 0][fs entry not found
%ProgramFilesDir%\Microsoft SQL Server\90\Tools\binn\foobar][fs entry not found
%ProgramFilesDir%\Microsoft SQL Server\90\Tools\binn]
*** FindFirstFileW 'c:\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system probe
c:\bin\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'c:\bin\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing in view
0][fs entry not found %drive_c%\bin\foobar][fs entry not found %drive_c%\bin]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\Tools\WinNT\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\Program Files\Microsoft Visual
Studio\Common\Tools\WinNT\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\Tools\WinNT\foobar' ->
INVALID_HANDLE_VALUE *** failed [FS missing in view 0][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\Tools\WinNT\foobar][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\Tools\WinNT]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\Program Files\Microsoft Visual
Studio\Common\MSDev98\Bin\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\foobar' ->
INVALID_HANDLE_VALUE *** failed [FS missing in view 0][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\MSDev98\Bin\foobar][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\MSDev98\Bin]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\Tools\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\Program Files\Microsoft Visual
Studio\Common\Tools\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\Common\Tools\foobar' ->
INVALID_HANDLE_VALUE *** failed [FS missing in view 0][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\Tools\foobar][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\Common\Tools]
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\VC98\bin\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\Program Files\Microsoft Visual
Studio\VC98\bin\foobar.* -> ffffffffh][no virtual or system matches]
```

```
*** FindFirstFileW 'C:\Program Files\Microsoft Visual Studio\VC98\bin\foobar' ->
INVALID_HANDLE_VALUE *** failed [FS missing in view 0][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\VC98\bin\foobar][fs entry not found
%ProgramFilesDir%\Microsoft Visual Studio\VC98\bin]
```

As you can see, the "potential errors" did a good job of highlighting possible areas where the application is failing.

Digging Deeper

```
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe
```

Let's suppose we want to find out why `cmd.exe` is probing the location `c:\test\cmd_test\bin` We can search the log for this line of text using the *log entry number* and find out what is occurring before this call. In the **bold** excerpts below you can two possible places where `cmd.exe` obtained the path `c:\test\cmd_test`. The first is by calling GetCurrentDirectoryW, and the second is from it calling GetFullPathNameW with "." as the path specified. In both cases, this returns the path for the current working directory—so now we know exactly how `cmd.exe` is obtaining this path. You can even see in the log file how `cmd.exe` creates the `c:\test\cmd_test\bin>` prompt—it does this by querying the environment variable "PROMPT" which returns "PG" and then uses the API function WriteConsoleW to print the prompt to the screen after internally expanding "PG" to `c:\test\cmd_test\bin>`

```
000824 0a88 cmd.exe :4ad0697a<-ADVAPI32.dll:77dd038f FormatMessageW ->DWORD=29h
(OUT LPWSTR lpBuffer=*4AD38BA0h->L"(C) Copyright 1985-2001 Microsoft Corp.\0Dh\0Ah")
000825 0a88 cmd.exe :4ad069d1->ADVAPI32.dll:77dd038f FormatMessageW (IN DWORD
dwFlags=1800h, IN LPCVOID lpSource=*0h, IN DWORD dwMessageId=2334h, IN DWORD
dwLanguageId=0h, IN DWORD nSize=2000h, IN *Arguments=*13DD40h->...)
000826 0a88                      FormatMessageW
FORMAT_MESSAGE_FROM_HMODULE FORMAT_MESSAGE_FROM_SYSTEM
line_width=unlimited lpSource=0x0, dwMessageId=0x2334, dwLanguageId=0x0
                                 -> 0x29 ((C) Copyright 1985-2001 Microsoft Corp.
 )
000827 0a88 cmd.exe  :4ad069d1<-ADVAPI32.dll:77dd038f FormatMessageW ->DWORD=29h
(OUT LPWSTR lpBuffer=*4AD38BA0h->L"(C) Copyright 1985-2001 Microsoft Corp.\0Dh\0Ah")
000828 0a88 cmd.exe  :4ad08d01->kernel32.dll:7c835484 WriteConsoleW (IN HANDLE
hConsoleOutput=7h, IN const *lpBuffer=*4AD38BA0h, IN DWORD nNumberOfCharsToWrite=29h, IN
LPVOID lpReserved=*0h)
000829 0a88 cmd.exe  :4ad08d01<-kernel32.dll:7c835484 WriteConsoleW ->BOOL=1h (OUT
LPDWORD lpNumberOfCharsWritten=*13DD24h->29h)
000830 0a88 cmd.exe  :4ad048f4->msctfime.ime:755c039b GetModuleHandleW (IN LPCWSTR
lpModuleName=*4AD0498Ch->L"KERNEL32.DLL")
000831                           0a88 GetModuleHandleW 'KERNEL32.DLL' -> 7c800000
000832 0a88 cmd.exe  :4ad048f4<-msctfime.ime:755c039b GetModuleHandleW -
->HMODULE=7c800000h ()
000833 0a88 cmd.exe  :4ad04907->AcGenral.DLL:6f880364 GetProcAddress (IN HMODULE
hModule=7c800000h, IN LPCSTR lpProcName=*4AD04980h->"CopyFileExW")
000834 0a88                      GetProcAddress
mod=7c800000/C:\WINDOWS\system32\kernel32.dll () 'CopyFileExW' -> 7feb1fcf
000835 0a88 cmd.exe  :4ad04907<-AcGenral.DLL:6f880364 GetProcAddress -
->FARPROC=*7FEB1FCFh ()
000836 0a88 cmd.exe  :4ad04919->AcGenral.DLL:6f880364 GetProcAddress (IN HMODULE
hModule=7c800000h, IN LPCSTR lpProcName=*4AD0496Ch->"IsDebuggerPresent")
000837 0a88                      GetProcAddress
mod=7c800000/C:\WINDOWS\system32\kernel32.dll () 'IsDebuggerPresent' -> 7fec0dfa
000838 0a88 cmd.exe :4ad04919<-AcGenral.DLL:6f880364 GetProcAddress -
->FARPROC=*7FEC0DFAh ()
000839 0a88 cmd.exe  :4ad0492b->AcGenral.DLL:6f880364 GetProcAddress (IN HMODULE
hModule=7c800000h, IN LPCSTR lpProcName=*4AD04954h->"SetConsoleInputExeNameW")
000840 0a88                      GetProcAddress
mod=7c800000/C:\WINDOWS\system32\kernel32.dll () 'SetConsoleInputExeNameW' -> 7fe90c21
000841 0a88 cmd.exe  :4ad0492b<-AcGenral.DLL:6f880364 GetProcAddress -
>FARPROC=*7FE90C21h ()
000842 0a88 cmd.exe  :4ad02c97->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=3h)
000843 0a88                     GetFileType 3 -> 0x2
000844 0a88 cmd.exe  :4ad02c97<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000845 0a88 cmd.exe  :4ad02cc0->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF6h)
```

Beta

```
000846 0a88 cmd.exe  :4ad02cc0<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=3h ()
000847 0a88 cmd.exe  :4ad02ccd->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=3h)
000848 0a88 cmd.exe  :4ad02ccd<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DDCCh->A7h)
000849 0a88 cmd.exe  :4ad05b74->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=7h)
000850 0a88                  GetFileType 7 -> 0x2
000851 0a88 cmd.exe  :4ad05b74<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000852 0a88 cmd.exe  :4ad05b9d->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF5h)
000853 0a88 cmd.exe  :4ad05b9d<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=7h ()
000854 0a88 cmd.exe  :4ad05baa->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=7h)
000855 0a88 cmd.exe  :4ad05baa<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DA80h->3h)
000856 0a88 cmd.exe  :4ad05dec->kernel32.dll:7c835484 WriteConsoleW (IN HANDLE
hConsoleOutput=7h, IN const *lpBuffer=*4AD38BA0h, IN DWORD nNumberOfCharsToWrite=2h, IN
LPVOID lpReserved=*0h)
000857 0a88 cmd.exe  :4ad05dec<-kernel32.dll:7c835484 WriteConsoleW ->BOOL=1h (OUT
LPDWORD lpNumberOfCharsWritten=*13DAACh->2h)
000858 0a88 cmd.exe  :4ad01ba8->USERENV.dll :769c03b9 GetEnvironmentVariableW (IN
LPCWSTR lpName=*4AD34624h->L"PROMPT", IN DWORD nSize=2000h)
000859 0a88                  GetEnvironmentVariable PROMPT -> $P$G
000860 0a88 cmd.exe  :4ad01ba8<-USERENV.dll :769c03b9 GetEnvironmentVariableW -
>DWORD=4h (OUT LPWSTR lpBuffer=*4AD2BA20h->L"$P$G")
000861 0a88 cmd.exe  :4ad01580->USERENV.dll :769c0396 GetCurrentDirectoryW (IN DWORD
nBufferLength=104h)
000862 0a88                  GetCurrentDirectoryW -> 0x14 (C:\test\cmd_test\bin)
000863 0a88 cmd.exe  :4ad01580<-USERENV.dll :769c0396 GetCurrentDirectoryW ->DWORD=14h
(OUT LPWSTR lpBuffer=*4AD34400h->L"C:\test\cmd_test\bin")
000864 0a88 cmd.exe  :4ad05b74->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=7h)
000865 0a88                  GetFileType 7 -> 0x2
000866 0a88 cmd.exe  :4ad05b74<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000867 0a88 cmd.exe  :4ad05b9d->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF5h)
000868 0a88 cmd.exe  :4ad05b9d<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=7h ()
000869 0a88 cmd.exe  :4ad05baa->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=7h)
000870 0a88 cmd.exe  :4ad05baa<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DA84h->3h)
000871 0a88 cmd.exe  :4ad05dec->kernel32.dll:7c835484 WriteConsoleW (IN HANDLE
hConsoleOutput=7h, IN const *lpBuffer=*4AD2B1E0h, IN DWORD nNumberOfCharsToWrite=15h, IN
LPVOID lpReserved=*0h)
000872 0a88 cmd.exe  :4ad05dec<-kernel32.dll:7c835484 WriteConsoleW ->BOOL=1h (OUT
LPDWORD lpNumberOfCharsWritten=*13DAB0h->15h)
000873 0a88 cmd.exe  :4ad0bf00->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=3h)
000874 0a88                  GetFileType 3 -> 0x2
000875 0a88 cmd.exe  :4ad0bf00<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000876 0a88 cmd.exe  :4ad02c97->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=3h)
000877 0a88                  GetFileType 3 -> 0x2
000878 0a88 cmd.exe  :4ad02c97<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000879 0a88 cmd.exe   :4ad02cc0->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF6h)
000880 0a88 cmd.exe   :4ad02cc0<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=3h ()
000881 0a88 cmd.exe   :4ad02ccd->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=3h)
000882 0a88 cmd.exe   :4ad02ccd<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DD50h->A7h)
000883 0a88 cmd.exe   :4ad02c97->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=3h)
000884 0a88                  GetFileType 3 -> 0x2
000885 0a88 cmd.exe   :4ad02c97<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
000886 0a88 cmd.exe   :4ad02cc0->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF6h)
000887 0a88 cmd.exe   :4ad02cc0<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=3h ()
000888 0a88 cmd.exe   :4ad02ccd->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=3h)
000889 0a88 cmd.exe   :4ad02ccd<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DD50h->A7h)
```

```
000890 0a88 cmd.exe   :4ad0b9d4->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF5h)
000891 0a88 cmd.exe   :4ad0b9d4<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=7h ()
000892 0a88 cmd.exe   :4ad0ba16->kernel32.dll:7c81bc2b GetConsoleScreenBufferInfo (IN
HANDLE hConsoleOutput=7h)
000893 0a88 cmd.exe   :4ad0ba16<-kernel32.dll:7c81bc2b GetConsoleScreenBufferInfo ->BOOL=1h
(OUT PCONSOLE_SCREEN_BUFFER_INFO lpConsoleScreenBufferInfo=*13DD08h->struct
{COORD dwSize=struct {SHORT X=50h, SHORT Y=12Ch}, COORD dwCursorPosition=struct
{SHORT X=15h, SHORT Y=5h}, WORD wAttributes=7h, SMALL_RECT srWindow=struct {SHORT
Left=0h, SHORT Top=0h, SHORT Right=4Fh, SHORT Bottom=18h}, COORD
dwMaximumWindowSize=struct {SHORT X=50h, SHORT Y=53h}})
000894 0a88 cmd.exe  :4ad0ba71->kernel32.dll:7c871a6c ReadConsoleW (IN HANDLE
hConsoleInput=3h, IN DWORD nNumberOfCharsToRead=2000h, IN LPVOID lpReserved=*13DD20h)
001518 0a88 cmd.exe  :4ad0ba71<-kernel32.dll:7c871a6c ReadConsoleW ->BOOL=1h (OUT
LPVOID lpBuffer=*4AD2FAE0h, OUT LPDWORD lpNumberOfCharsRead=*13DD70h->8h)
001519 0a88 cmd.exe  :4ad02c97->ole32.dll :774e03f0 GetFileType (IN HANDLE hFile=3h)
001520 0a88                    GetFileType 3 -> 0x2
001521 0a88 cmd.exe  :4ad02c97<-ole32.dll :774e03f0 GetFileType ->DWORD=2h ()
001522 0a88 cmd.exe  :4ad02cc0->kernel32.dll:7c812f39 GetStdHandle (IN DWORD
nStdHandle=FFFFFFF6h)
001523 0a88 cmd.exe  :4ad02cc0<-kernel32.dll:7c812f39 GetStdHandle ->HANDLE=3h ()
001524 0a88 cmd.exe  :4ad02ccd->kernel32.dll:7c81af14 GetConsoleMode (IN HANDLE
hConsoleHandle=3h)
001525 0a88 cmd.exe  :4ad02ccd<-kernel32.dll:7c81af14 GetConsoleMode ->BOOL=1h (OUT
LPDWORD lpMode=*13DD50h->A7h)
001526 0a88 cmd.exe  :4ad0bb9c->kernel32.dll:7c81b18f GetConsoleOutputCP ()
001527 0a88 cmd.exe  :4ad0bb9c<-kernel32.dll:7c81b18f GetConsoleOutputCP ->UINT=1B5h ()
001528 0a88 cmd.exe  :4ad0bbad->kernel32.dll:7c812e76 GetCPInfo (IN UINT CodePage=1B5h)
001529 0a88 cmd.exe  :4ad0bbad<-kernel32.dll:7c812e76 GetCPInfo ->BOOL=1h (OUT LPCPINFO
lpCPInfo=*4AD33BA0h->struct {UINT MaxCharSize=1h, char[2] DefaultChar=['?', '\00h'], char[12]
LeadByte=['\00h', '\00h', '\00h', '\00h', '\00h', '\00h', '\00h', '\00h', '\00h', '\00h',
'\00h']})
001530 0a88 cmd.exe  :4ad01680->kernel32.dll:7c81b258 SetThreadUILanguage (== no prototype
available ==)
001531 0a88 cmd.exe  :4ad01680<-kernel32.dll:7c81b258 SetThreadUILanguage (== no prototype
available ==)
001532 0a88 cmd.exe  :4ad01b0d->kernel32.dll:7c80ac0f SetErrorMode (IN UINT uMode=0h)
001533 0a88 cmd.exe  :4ad01b0d<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001534 0a88 cmd.exe  :4ad01b13->kernel32.dll:7c80ac0f SetErrorMode (IN UINT uMode=1h)
001535 0a88 cmd.exe   :4ad01b13<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001536 0a88 cmd.exe  :4ad01b24->IMM32.DLL :7639039b GetFullPathNameW (IN LPCWSTR
lpFileName=*1638C0h->L".", IN DWORD nBufferLength=208h)
001537 0a88                    GetFullPathNameW . -> 20 (buf=C:\test\cmd_test\bin,
file_part=bin)
001538 0a88 cmd.exe  :4ad01b24<-IMM32.DLL :7639039b GetFullPathNameW ->DWORD=14h
(OUT LPWSTR lpBuffer=*163D60h->L"C:\test\cmd_test\bin", OUT *lpFilePart=*13D8D4h-
>*163D82h->L"bin")
001539 0a88 cmd.exe   :4ad01b29->kernel32.dll:7c80ac0f SetErrorMode (IN UINT uMode=0h)
001540 0a88 cmd.exe   :4ad01b29<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=1h ()
001541 0a88 cmd.exe   :4ad01ba8->USERENV.dll :769c03b9 GetEnvironmentVariableW (IN
LPCWSTR lpName=*4AD34618h->L"PATH", IN DWORD nSize=2000h)
001542 0a88                    GetEnvironmentVariable PATH ->
C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;c:\program
files\subversion\bin;c:\Program Files\Microsoft SQL Server\90\Tools\binn\;c:\bin;C:\Program
Files\Microsoft Visual Studio\Common\Tools\WinNT;C:\Program Files\Microsoft Visual
Studio\Common\MSDev98\Bin;C:\Program Files\Microsoft Visual Studio\Common\Tools;C:\Program
Files\Microsoft Visual Studio\VC98\bin
001543 0a88 cmd.exe   :4ad01ba8<-USERENV.dll :769c03b9 GetEnvironmentVariableW -
>DWORD=173h (OUT LPWSTR lpBuffer=*4AD2BA20h-
->L"C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;c:\program files\su.. ")
001544 0a88 cmd.exe   :4ad01ba8->USERENV.dll :769c03b9 GetEnvironmentVariableW (IN
LPCWSTR lpName=*4AD34608h->L"PATHEXT", IN DWORD nSize=2000h)
001545 0a88                    GetEnvironmentVariable PATHEXT ->
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
001546 0a88 cmd.exe   :4ad01ba8<-USERENV.dll :769c03b9 GetEnvironmentVariableW -
->DWORD=30h (OUT LPWSTR lpBuffer=*4AD2BA20h-
-> L".COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH")
```

Retail

```
001547 0a88 cmd.exe   :4ad02aaa->kernel32.dll:7c80b2d0 GetDriveTypeW (IN LPCWSTR
lpRootPathName=*13D8C4h->L"C:\")
001548 0a88 cmd.exe   :4ad02aaa<-kernel32.dll:7c80b2d0 GetDriveTypeW ->UINT=3h ()
001549 0a88 cmd.exe   :4ad01b5f->USERENV.dll :769c03fa FindFirstFileW (IN LPCWSTR
lpFileName=*1638C0h->L"C:\test\cmd_test\bin\foobar.*")
001550 0a88                   FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no
virtual or system matches]
```

sbmerge

sbmerge (Sandbox Merge)

Sandbox Merge (`sbmerge.exe`) is used to merge runtime changes recorded in the application sandbox back into a VMware Project North Star (Thinstall) project.

A typical work-flow for using sbmerge looks like this:

1 Use Setup Capture to create a VMware Project North Star (Thinstall) project

2 Build Thinstalled Application using `build.bat`

3 Run Thinstalled Application, customize user settings and virtual environment -> changes go to sandbox

4 Run sbmerge to merge registry and file system changes from sandbox into VMware Project North Star (Thinstall) project

5 Build updated Thinstalled Application using `build.bat`

6 Deploy updated Thinstalled Application (includes customizations from step 3)

Usage:

```
sbmerge.exe Print [OptionalParameters]
sbmerge.exe Apply [Optional Parameters]
```

Optional Parameters:

```
[-ProjectDir PathToProject] [-SandboxDir PathToSandbox]
[-Quiet] [-Exclude ExcludeFile.ini]
```

Main Operation

`Print` will display sandbox changes. This is a non-destructive operation, no modifications will be made to the sandbox or original project.

`Apply` will merge changes from the sandbox into the specified project. The project registry and filesystem will be updated to reflect changes from the sandbox. The sandbox directory will be deleted when this operation completes.

Optional Parameters

`-ProjectDir` specifies project directory, default = current directory

`-SandboxDir` specifies sandbox location, default = active sandbox

`-Quiet` will not print progess messages

`-Exclude` specifies an exclusion file (which files & keys to ignore)

Example Usage

```
c:\myproject> sbmerge Print
```

Prints changes that are recorded in the sandbox

```
c:\myproject> sbmerge Apply
```

Merges the changes from the sandbox into the project directory located at `c:\myproject sbmerge` uses the normal search order for sandbox probing

Beta

```
c:\somedir> sbmerge Apply –ProjectDir c:\myproject

c:\somedir> sbmerge Apply –ProjectDir c:\myproject –SandboxDir "c:\documents and
               settings\username\application data\thinstall\myproject"
```

ThinReg

ThinReg (Thinstall Registration Tool)

ThinReg (`ThinReg.exe`) is used to register VMware Project North Star (Thinstall) packages on a PC, including:

- Creating StartMenu & Desktop Shortcuts
- Setting up File type associations
- Adding uninstall information that can be executed from the system Control Panel
- Unregistering previously registered packages

Usage:

```
ThinReg.exe [Optional Parameters] PackageName.exe [Package2.exe] [PackagesByWildcard]
```

Optional Parameters include:

`/a, /allusers`

Register package for all users (by default package is registered only for the currently logged in user)

`/q, /quiet`

Don't output any information

`/u, /unregister, /uninstall`

Unregister package. If needed, this will also remove the software from the Add/Remove Programs control panel applet.

`/r, /reregister`

Re-register package. Normally, ThinReg will detect if a package is already registered and will skip it if it is. By using the /r option, you can force ThinReg to re-register the package

`/k, /keepunauthorized, /keep`

If ThinReg detects a registration for a package for which you're no longer authorized (you are no longer part of the PermittedGroups specified in the packages `Package.ini`) it will remove the registration info for that package. By specifying this option, ThinReg won't remove the registration info.

`/noarp`

Don't create an entry in the Add/Remove Programs control panel applet

`/norelaunch`

This option has an effect on Vista only. ThinReg will be launched on Vista without elevated privileges. That means that standard users will be able to invoke ThinReg without a UAC pop-up. When ThinReg detects that it does need more privileges (those privileges are required for the /allusers option), it will relaunch itself as a privileged process, causing a UAC pop-up to show. The /norelaunch option will prevent the relaunch and instead the registration will just fail.

The standard usage of ThinReg would be to register a number of packages stored in a directory (probably on a file server) during login processing: `ThinReg \\server\share\dir\*.exe`. This will automatically take care of PermittedGroups specifications, registering and unregistering packages as needed. It is also quite fast. During testing, we found that first-time registration of the Office 2003 suite took 0.55 sec. Second-time registration (where ThinReg didn't need to perform any action) took only 0.13 sec. By contrast, /reregister is relatively expensive at 5.40 sec so you should probably not use that in a login script.

Beta

Example usage:

This will register just Winword for the currently logged in user

```
ThinReg.exe "\\server\share\Microsoft Office 2003 Winword.exe"
```

This will register just Winword for all users on the system (requires Admin Rights)

```
ThinReg.exe /a "\\server\share\Microsoft Office 2003 Winword.exe"
```

This will register all Office applications in the specified directory for the currently logged in user

```
ThinReg.exe "\\server\share\Microsoft Office *.exe"
```

This will unregister register just Winword for the current user. Registrations for e.g. Excel won't be affected

```
ThinReg.exe /u "\\server\share\Microsoft Office 2003 Winword.exe"
```

This will unregister all Office applications for the current user. Will also remove the Add/Remove Programs entry

```
ThinReg.exe /u "\\server\share\Microsoft Office *.exe"
```

NOTE Command-line unregistration is supported in VMware Project North Star (Thinstall) 3.210 and higher

snapshot

Snapshot

`Snapshot.exe` can be used for:

- Saving the state of a PC's filesystem and registry

  Example:

  ```
  snapshot C:\data.snapshot snapshot
  C:\data.snapshot C:\ HKEY_LOCAL_MACHINE
  ```

- Comparing two previously recorded states

  Example:

  ```
  snapshot C:\start.snapshot –diffprint C:\end.snapshot
  ```

- Printing contents of a saved state

  Example:

  ```
  snapshot C:\start.snapshot –print
  ```

- Generating a VMware Project North Star (Thinstall) project by comparing two previously saved states

  Example:

  ```
  snapshot C:\start.snapshot –SuggestProject C:\end.snapshot C:\project.ini
  snapshot C:\project.ini –GenerateProject
  ```

Usage:

```
snapshot.exe SnapshotFileName.snapshot [–Config ConfigFile.ini][BaseDir1][BaseDir2][BaseReg1]
snapshot.exe Snap1.snapshot Snap2.snapshot OutDir [–Config ConfigFile.ini]
snapshot.exe Snap1.snapshot –SuggestProject Snap2.snapshot OutputTemplate.ini
snapshot.exe Template.ini –GenerateProject OutDir [–Config ConfigFile.ini]
snapshot.exe SnapshotFileName.snapshot –Print
```

Example .bat file to generate a VMware Project North Star (Thinstall) project using a start and ending snapshot

```
snapshot c:\start.snapshot
echo Please install application and press ENTER when installation is complete
pause
snapshot C:\end.snapshot s
napshot C:\start.snapshot –SuggestProject c:\end.snapshot c:\SuggestedProject.ini
```

Beta

rem at this point a GUI app can look at SuggestedProject and ask the user what EXEs they want to be accessible from `package.ini`)

```
snapshot C:\SuggestedProject.ini –GenerateProject c:\ProjectLocation
del C:\start.snapshot
del C:\end.snapshot
del C:\SuggestedProject.ini
```

dll_dump

```
C:\Program Files\Thinstall.VS>dll_dump.exe
```

Usage:

`dll_dump ADDRESS` (show DLL & process which has this address loaded)

`dll_dump SUBSTRING` (shows DLLs loaded by VMware Project North Star (Thinstall) processes where name matches `SUBSTRING`)

`dll_dump *` (shows DLLs loaded by all VMware Project North Star (Thinstall) processes)

`dll_dump –fp` (show DLL full path, not just filenames)

`dll_dump ADDRESS SUBSTRING` (only show processes matching `SUBSTRING` where `ADDRESS` is loaded)

One of the most useful purposes for `dll_dump` is to list all running Thinstalled applications on a PC.

If you use a spy program like Process Explorer on a Thinstalled app, you will not see DLLs which are loaded by VMware Project North Star (Thinstall) since they have been virtualized and Windows does not really know they exist. Likewise, if you attach a debugger to a running VMware Project North Star (Thinstall) process, the debugger will not be aware of virtual DLLs. If you are investigating code running at a specific address, you can use dll_dump to convert this address into a virtual DLL name and base address.

Using log monitor, you can generate a trace and convert this to text format. In the report near the end, you'll find a section labeled:

```
– Modules loaded –
```

This section lists all DLLs that were loaded by the application over the course of its execution history.

DLLs that are described as "SYSTEM_LOADED" are loaded by Windows from the host PC; these will include all the basic OS DLLs like `kernel32.dll`

DLLs that are described as "MEMORY_MAPPED_ANON" are loaded by VMware Project North Star (Thinstall) and completely isolated from Windows.

For Adobe Reader, you should see something like this:

```
PRELOADED_BY_SYSTEM 00400000–00410000, C:\Program Files\Adobe\Acrobat 7.0\Reader\AcroRd32.exe
PRELOADED_MAP 00400000–00410000, C:\Program Files\Adobe\Acrobat 7.0\Reader\AcroRd32.exe
SYSTEM_LOADED 00400000–00410000, C:\thinstest\1072\Adobe Reader 7.0\bin\Adobe.exe
SYSTEM_LOADED 77dd0000–77e6b000, C:\WINDOWS\system32\ADVAPI32.dll SYSTEM_LOADED
76fd0000–7704f000, C:\WINDOWS\system32\CLBCATQ.DLL
SYSTEM_LOADED 5d090000–5d127000, C:\WINDOWS\system32\comctl32.dll ...

MEMORY_MAPPED_ANON 05000000–05085000, C:\Program Files\Adobe\Acrobat 7.0\Reader\ACE.dll

MEMORY_MAPPED_ANON 03000000–038c9000, C:\Program Files\Adobe\Acrobat 7.0\Reader\AcroRd32.dll
MEMORY_MAPPED_ANON 06000000–061aa000, C:\Program Files\Adobe\Acrobat 7.0\Reader\AGM.dll
MEMORY_MAPPED_ANON 07000000–0701b000, C:\Program Files\Adobe\Acrobat 7.0\Reader\BIB.dll
MEMORY_MAPPED_ANON 7c3a0000–7c41b000, C:\WINDOWS\system32\MSVCP71.dll
```

Beta

VREGTool

VREGTool is VMware Project North Star (Thinstall)'s virtual registry compiler and exporter

Command Line Usage:

NOTE All parameters are case-insensitive.

```
VREGTOOL regfile.tvr ExportDir output_directory [HKEY_LOCAL_MACHINE\Software]
VREGTOOL regfile.tvr ImportDir input_directory
VREGTOOL regfile.tvr ImportReg regedit.reg [-Merged|-WriteCopy|-Full] [-NoReplace][-NoMacros]
VREGTOOL regfile.tvr ExportReg filename.reg [HKEY_LOCAL_MACHINE\Software]
VREGTOOL regfile.tvr PrintKeys [HKEY_LOCAL_MACHINE\Software] [-ShowValues] [-ShowData]
              [-ExpandMacros]
VREGTOOL regfile.tvr PrintStats
VREGTOOL regfile.tvr SysCompare [HKEY_LOCAL_MACHINE\Software] [-Exact]
VREGTOOL regfile.tvr DelSubkey HKEY_LOCAL_MACHINE\Software [-NoMark]
```

Exporting registry data to VMware Project North Star (Thinstall) registry directory format

```
VREGTOOL regfile.tvr ExportDir output_directory [HKEY_LOCAL_MACHINE\Software]
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

`output_directory` : Output directory where registry data will be written

`HKEY_LOCAL_MACHINE\Software` : You can optionally specify a registry subtree to export. If no subtree is specified, VMware Project North Star (Thinstall) will export the entire contents of the `.tvr` file. If the specified registry subkey has a space in the name, you should specify the key using quotes like this: `"HKEY_LOCAL_MACHINE\Software\Key with space"`

Importing registry data from VMware Project North Star (Thinstall) registry directory format

```
VREGTOOL regfile.tvr ImportDir input_directory
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

`input_directory` : Input directory where registry data will be read from

Example:

```
vregtool c:\tmp\test.tvr ImportDir
              "C:\jc\code\VOS2\thinstall\setup_capture\Debug\Captures\05-07-2006 01.14"
```

Importing registry data from regedit format

```
VREGTOOL regfile.tvr ImportReg regedit.reg [-Merged|-WriteCopy|-Full] [-NoReplace][-NoMacros]
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

`regedit.reg` : The `.reg` file to import, entries imported will be added to the specified `.tvr` file. This file can be in REGEDIT 4.0 (ansi text) or 5.0 (unicode text) format.

Isolation Mode options (if no isolation mode is specified WriteCopy is used)

`Merged` : specifies that registry keys which do not already exist will have isolation mode set to Merged. (Chapter 4, "Isolation Modes," on page 61)

`WriteCopy`: specifies that registry keys which do not already exist will have isolation mode set toWriteCopy. (Chapter 4, "Isolation Modes," on page 61)

`Full` : specifies that registry keys which do not already exist will have isolation mode set to Full.

`NoReplace` : Do not replace or modify existing registry values. If this option is not selected, when a `.tvr` already contains a value specified in the `.reg` file, this value will be overwritten with the value specified in the `.reg` file.

Beta

NoMacros : Do not perform macro substitution for registry values that contain paths to short pathname or shell folders. When this flag is not set, the values contained in .tvr will be replaced with macro versions of paths. For example, if the .reg file contains a string registry value of C:\windows\system32\kernel32.dll, the .tvr file will contain the value %systemsystem%\kernel32.dll. When the application requests the value of this registry key, it will receive the value C:\windows\system\kernel32.dll when running on Windows 98,ME,and XP+. If the application runs on NT or 2k, it will receive the value C:\winnt\system32\kernel32.dll. Usually macro substitution is preferable; however, in some unusual cases, you may want to disable this and store a hard-coded path in the registry.

Exporting registry data to regedit format

```
VREGTOOL regfile.tvr ExportReg filename.reg [HKEY_LOCAL_MACHINE\Software]
```

regfile.tvr : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

regedit.reg : The .reg file to export to. The file produced will be in REGEDIT 5.0 format (unicode text).

HKEY_LOCAL_MACHINE\Software : You can optionally specify a registry subtree to export. If no subtree is specified, vregtool will export the entire contents of the .tvr file. If the specified registry subkey has a space in the name, your should specify the key using quotes like this: "HKEY_LOCAL_MACHINE\Software\Key with space"

NOTE Special note: when exporting to .reg format, some information is lost, so exporting and reimporting may cause a loss of information. Information that can be lost includes:

1. Filename macros are expanded on export; if they are not converted back to macros on import information could be lost.
2. Each Registry subkey in .tvr format has a specified isolation mode, because .reg format does not have a concept of isolation modes or meta data for subkeys this information is lost.
3. Registry values that cannot be represented in .reg format; for example, a key which is REG_SZ cannot have more than one NULL character in .reg format. In such a case, the registry value data will be prematurely truncated in .reg format. Another example where .reg files cannot represent values accurately is a REG_SZ value which is not null-terminated. For these reasons, it is preferable to export registry data to VMware Project North Star (Thinstall) registry directory format and then re-import into the .tvr file, when you need to perform some processing on the data.

Example:

```
vregtool c:\tmp\test.tvr ExportReg c:\tmp\test.reg "HKEY_CURRENT_USER\Software\Adobe\Save For Web
              3.0"
```

Listing all registry keys in a Thinstall .trv file

```
VREGTOOL regfile.tvr PrintKeys [HKEY_LOCAL_MACHINE\Software] [–ShowValues] [–ShowData]
[–ExpandMacros]
```

regfile.tvr : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

HKEY_LOCAL_MACHINE\Software : You can optionally specify a registry subtree to print. If no subtree is specified, vregtool will print the entire contents of the .tvr file. If the specified registry subkey has a space in the name, your should specify the key using quotes like this: "HKEY_LOCAL_MACHINE\Software\Key with space"

ShowValues : Optionally prints the names of all virtual values contained in virtual subkeys

ShowData : Optionally prints the data associated with each virtual value

ExpandMacros : Optionally expands macros contained in registry values & data before printing

Example:

```
vregtool c:\tmp\test.tvr PrintKeys "HKEY_CURRENT_USER\Software\Adobe\Save For Web 3.0"
```

This operation will print all the virtual registry keys contained in a .tvr file to the console.

Beta

Listing diagnostic information about a Thinstall.trv file

```
VREGTOOL regfile.tvr PrintStats
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

This option is used mainly for diagnosing issues with .tvr files

Example:

```
vregtool c:\tmp\test.tvr PrintStats
```

Comparing virtual registry information with host PC registry information

```
VREGTOOL regfile.tvr SysComapre [HKEY_LOCAL_MACHINE\Software] [–Exact]
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

`HKEY_LOCAL_MACHINE\Software` : Optionally specifies the registry subkey to begin comparing. If this is specified, only subkeys at this level and below will be considered in the comparison.

`–Exact`: Optionally specifies that system comparison should print out all differences between the virtual registry and system registry. If this option is not specified, vregtool will not print out system registry keys that do not exist in the virtual registry if the virtual registry subkey is set to "merged" or "writecopy" isolation mode (Chapter 4, "Isolation Modes," on page 61) (but will print differences for subkeys with isolation mode set to "Full").

This option is useful in finding differences between a working system registry and virtual registry file.

Example:

```
vregtool c:\tmp\test.tvr SysCompare "HKEY_CURRENT_USER\Software\Adobe"
```

Deleting a registry subkey

```
VREGTOOL regfile.tvr DelSubkey HKEY_LOCAL_MACHINE\Software [–NoMark]
```

`regfile.tvr` : Datafile in VMware Project North Star (Thinstall) Virtual Registry file format

`HKEY_LOCAL_MACHINE\Software` : Specified the registry subkey to delete. If the specified registry subkey has a space in the name, your should specify the key using quotes like this:
`"HKEY_LOCAL_MACHINE\Software\Key with space"`

`NoMark` : Optionally specifies that the subkey should be deleted if it exists, but not marked as deleted. When a subkey is marked as deleted, applications running under VMware Project North Star (Thinstall) will not see the specified subkey even if it exists on the local system.

NOTE When `–NoMark` is not used, vregtool will mark the subkey as deleted whether or not it originally exists in the `.tvr` file.

Example:

```
vregtool c:\tmp\test.tvr DelSubkey "HKEY_CURRENT_USER\Software\Adobe\Save For Web 3.0"
```

TLink

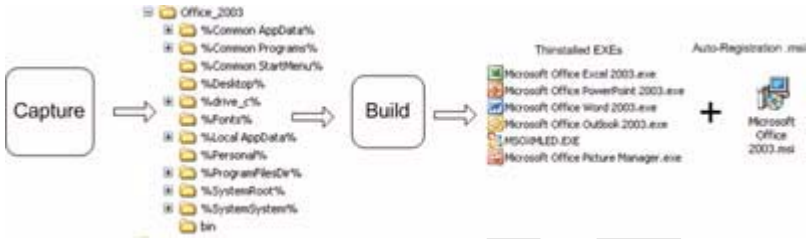
TLINK package.ini [-OutDir OutputDirectory]

Beta

Beta

MSI Generation 7

VMware Project North Star (Thinstall) can optionally generate auto-registering .msi files during the build process.



When to consider building an MSI

You may consider building an MSI database when you need to deliver VMware Project North Star (Thinstall) packaged applications through traditional desktop management systems to remote locations and have shortcuts and filetype associations automatically created. .msi packages provide a very simply way to do this. Most systems including plain Active Directory Group Policy provide simple methods to push and execute .msi packages.

What is included in a MSI database created by VMware Project North Star (Thinstall)?

1 A copy of all the Thinstalled EXE files for a specific project

2 A copy of the ThinReg registration utility. ("ThinReg" on page 86)

What happens when a VMware Project North Star (Thinstall) created .msi package is executed on an end-user desktop?

1 The msi database will extract your Thinstalled EXE file(s) to `c:\program` files. You can use the optional configuration option MSIInstallDirectory ("MSIInstallDirectory" on page 106) to control where files are extracted to.

If you specify ALLUSERS="" to msiexec, or you create a .msi package that install's per-user by default using the configuration option MSIDefaultInstallAllUsers ("MSIDefaultInstallAllUsers" on page 105) then the Thinstalled EXE files will be extracted to the users `%AppData%` directory.

2 The msi install will execute ThinReg to register filetypes and create shortcuts. ("ThinReg" on page 86)

How to build an MSI database

1 Edit a package file generated by `package.ini` to specify "MSIFilename=<filename>.msi" ("MSIFilename" on page 106)

2 run `build.bat`

NOTE If MSIFilename ("MSIFilename" on page 106) is not specified, msi generation will be skipped during the build process.

By default, the generated database will install machine-wide. You can change this by setting the option

```
MSIDefaultInstallAllUsers=0
```

("MSIDefaultInstallAllUsers" on page 105)

This will create a database which will have a default of per-user installation.msiexec command-line parameters

Irrespective of what you specified at package build time, you can still override the type of installation at deploy time. For example, if you created the database with "MSIDefaultInstallAllUsers=1" ("MSIDefaultInstallAllUsers" on page 105), you can still force a per-user deployment by deploying using:

```
msiexec /i <database>.msi ALLUSERS=""
```

If you want to force a per-machine deployment you'd use:

```
msiexec /i <database>.msi ALLUSERS=1
```

To default to per-machine install for Administrators (belonging to the "Administrators" group) and per-user for non-Administrators, set the value of MSIDefaultInstallAllUsers to 2. ("MSIDefaultInstallAllUsers" on page 105)

When doing a per-machine deployment, the default installation directory is the localized equivalent of `"%ProgramFilesDir%\<InventoryName> (Thinstalled)`. For a per-user deploy, the default is `%AppData%\<InventoryName> (Thinstalled)`. In both cases, you can override the installation directory by passing an INSTALLDIR property to msiexec:

```
msiexec /i <database>.msi INSTALLDIR=C:\Mydir\Mypackage
```

Vista

For deployment on Vista, you need to indicate whether an installer needs elevated privileges or not. Normally, when doing a per-user install you won't need elevated privileges but when doing a per-machine install you will. You can specify whether the package needs elevated privileges by using the "MSIRequireElevatedPrivileges" option ("MSIRequireElevatedPri..." on page 108). If you're going to deploy the installer machine-wide, set the value to "1". This will result in UAC prompts (if you have them enabled), even if you force a per-user installation by using the ALLUSERS="" commandline option. By setting the option to "0", you won't get a UAC prompt but the install will fail if you then try to install it machine-wide.

Handling Upgrades

A MSI database contains a number of codes, among which a "ProductCode" and an "UpgradeCode". When rebuilding a package, it is recommended that the UpgradeCode is kept the same. This can be specified using the "MSIUpgradeCode" option ("MSIUpgradeCode" on page 108). You can change the ProductCode (via the MSIProductCode option) ("MSIProductCode" on page 107) if desired. The advantage of changing the ProductCode is that it is easier to install a newer version. If the ProductCode of the new version is the same as the ProductCode of the old version, the install will prompt you to remove the old version using Add/Remove Programs first. If the ProductCode is different, the install will silently uninstall the old version and then install the new version.

If you don't specify a MSIProductCode option ("MSIProductCode" on page 107), a different ProductCode will be generated for each build automatically.

Beta

Package.ini format

8

The `Package.ini` file contains one [BuildOptions] and one or more individual [Application.exe] sections.

The [BuildOptions] apply to all applications, and any options set in this section will be inherited by individual applications, unless the application section specifically overrides these settings.

General Options

`CompressionType`—Controls what type of compression is used ("CompressionType" on page 100)

`DisableTracing`—Prevents `.trace` file generation when Log Monitor is running ("DisableTracing" on page 101)

`UpgradePath`—Location to probe for new versions of the application ("UpgradePath" on page 114)

`AddPageExecutePermission`—Used to fix applications that don't work in DEP environments ("AddPageExecutePermi..." on page 98)

`NetRelaunch`—Controls whether to relaunch an application from the local disk when running from a net share or removable disk ("NetRelaunch" on page 108)

`AutoStartServices`—Determines if virtual services are automatically started when the first application starts ("AutoStartServices" on page 98)

`AutoShutdownServices`—Determines if virtual services are automatically stopped when the last application process exits ("AutoShutdownServices" on page 98)

`LogPath`—Controls where to store .trace files when logging ("LogPath" on page 105)

`VirtualDrives`—Specifies additional drive letters that should be available to the application at runtime ("VirtualDrives" on page 114)

Sandbox Control

`SandboxName`—Sets name of directory where sandbox is created and stored ("SandboxName" on page 111)

`SandboxPath`—Controls the path where VMware Project North Star (Thinstall) will create and look for a new sandbox by default ("SandboxPath" on page 112)

`InventoryName`—Optional string that is used for package indentification by inventory control systems ("InventoryName" on page 104)

`SandboxNetworkDrives`—Determines if network-mapped drives will have sandboxing applied ("SandboxNetworkDrives" on page 112)

`SandboxRemovableDisk`—Determines if removable disk will have sandboxing applied ("SandboxRemovableDisk" on page 112)

`RemoveSandboxOnExit`—Resets the application by deleting the sandbox when the last child process exits ("RemoveSandboxOnExit" on page 111)

Beta

Isolation & Virtualization Granuality

`ExternalCOMObjects`—Controls weither a specific COM object CLSID will be created by VMware Project North Star (Thinstall) or by Windows ("ExternalCOMObjects" on page 102)

`VirtualizeExternalOutOfProcessCOM`—Controls whether external out-of-process COM objects are run in the virtual environment ("VirtualizeExternalOut..." on page 114)

`ChildProcessEnvironmentDefault`—Determines if child processes are run in the virtual environment by default ("ChildProcessEnvironme..." on page 99)

`ChildProcessEnvironmentExceptions`—Enables exceptions as the default child execution policy ("ChildProcessEnvironme..." on page 99)

`DirectoryIsolationMode`—Controls default isolation for directories in package ("DirectoryIsolationMode" on page 101)

`ExternalDLLs`—Force some DLLs to be loaded by Windows ("ExternalDLLs" on page 103)

`IsolatedMemoryObjects`—Lists specific shared memory objects to isolate from other applications ("IsolatedMemoryObjects" on page 104)

`IsolatedSynchronizationObjects`—Lists specific synchronization objects to isolate from other applications ("IsolatedSynchronizatio..." on page 105)

`RegistryIsolationMode`—Controls default isolation for registry keys in package ("RegistryIsolationMode" on page 109)

MSI Generation

`MSIDefaultInstallAllUsers`—Set default installation mode of the MSI database ("MSIDefaultInstallAllUsers" on page 105)

`MSIFilename`—Enable generation of an MSI database and specify its filename ("MSIFilename" on page 106)

`MSIInstallDirectory`—Specify the final path component of the default installation directory ("MSIInstallDirectory" on page 106)

`MSIManufacturer`—Specify the manufacturer to put in the MSI database ("MSIManufacturer" on page 107)

`MSIProductCode`—Specify a product code for the MSI database ("MSIProductCode" on page 107)

`MSIProductVersion`—Specify a product version number for the MSI database ("MSIProductVersion" on page 107)

`MSIRequireElevatedPrivileges`—Mark the MSI database as "requires elevated privileges" ("MSIRequireElevatedPri..." on page 108)

`MSIUpgradeCode`—Specify an upgrade code for the MSI database ("MSIUpgradeCode" on page 108)

Access Control

`PermittedGroups`—Restricts usage of package to a specific set of Active Directory groups ("PermittedGroups" on page 110)

`AccessDeniedMsg`—Holds error message to display to user if they do not have permission to run package ("AccessDeniedMsg" on page 97)

Application-Specific Options

`Disabled`—Indicates that a build target is a placeholder and thus should not generate an EXE ("Disabled" on page 102)

`Icon`—Indicates which icon file to use for the generated `EXE` file ("Icon" on page 103)

`CommandLine`—Specify command line arguments for a shortcut executable ("CommandLine" on page 100)

`NoRelocation`—Strips relocation information from the resulting executable ("NoRelocation" on page 109)

Beta

`RetainAllIcons`—Indicates all of the source EXEsoriginal icons should be retained in the Thinstalled EXE files ("RetainAllIcons" on page 111)

`Source`—Points to the `EXE` file which will be initially loaded by VMware Project North Star (Thinstall) ("Source" on page 113)

`ReserveExtraAddressSpace`—Indicates how much extra address space needs to be reserved for the Thinstalled executable ("ReserveExtraAddress..." on page 110)

`Shortcut`—Points to the name of the VMware Project North Star (Thinstall) generated EXE which will host the package data ("Shortcut" on page 113)

`StripVersionInfo`—Removes all version information from the source EXE when building the target application ("StripVersionInfo" on page 113)

`WorkingDirectory`—Sets the current working directory before the application starts ("WorkingDirectory" on page 116)

`Version.XXXX`—Used to override EXE version strings or add new version strings ("Version.XXXX" on page 115)

Example package.ini file

[BuildOptions]

```
SandboxName=MainApp v1.0
```

("SandboxName" on page 111)

[Compression]

```
CompressionType=Fast
```

("CompressionType" on page 100)

[Isolation]

```
DirectoryIsolationMode=WriteCopy
```

("DirectoryIsolationMode" on page 101)

[MainApp.exe]

```
Source=%ProgramFiles%\Test\MainApp.exe
```

("Source" on page 113)

[Test.exe]

```
Source=%ProgramFiles%\Test\UtilityApp.exe
```

("Source" on page 113)

```
Shortcut=MainApp.exe
```

("Shortcut" on page 113)

AccessDeniedMsg

`AccessDeniedMsg`—Holds error message to display to user if they do not have permission to run package

Example

```
[BuildOptions]
PermittedGroups=Administrator;OfficeUsers
AccessDeniedMsg=You do not have permission to execute this application, please call support @
1-800-822-2992
```

This setting controls the string which is displayed to the user if they are not authorized to execute the application.

Beta

AddPageExecutePermi...

`AddPageExecutePermission`—Used to fix applications that don't work in DEP environments

Windows XP SP2, Windows Server 2003, and above have a feature called Data Execution Prevention which is designed to protect against some security exploits that occur with buffer overflows. Because this feature causes a number of compatibility issues, it is turned off by default on XP SP2 and it is possible to use an machine-specific opt-in or opt-out list of which applications to apply DEP protection to. Opt-in and opt-out policies can be difficult to manage when a large number of machines and applications are involved. This option instructs VMware Project North Star (Thinstall) to add Execution Permission to pages allocated by an application so that it can run on machines that have DEP protection enabled, without having to modify the opt-out list.

Examples

Disable some Data Execution protections for this particular application

```
[BuildOptions]
AddPageExecutionPermission=1
```

Don't mess with DEP protections (default)

```
[BuildOptions]
AddPageExecutionPermission=0
```

AutoShutdownServices

`AutoShutdownServices`—Controls whether to automatically shutdown virtual services when the last non-service process exits

By default, VMware Project North Star (Thinstall) will automatically shutdown virtual services when the last non-service-based child process exits. This option instructs VMware Project North Star (Thinstall) to keep virtual services running even when all other processes have exited. This option does not have any effect on non-virtual services.

Examples

Keep virtual services running when the application exits

```
[BuildOptions]
AutoShutdownServices=0
```

Stop virtual services when the last non-service application exits (default)

```
[BuildOptions]
AutoShutdownServices=1
```

See also: "AutoStartServices" on page 98

AutoStartServices

`AutoStartServices`—Controls wheter to automatically start virtual service when the first application starts

By default VMware Project North Star (Thinstall) will automatically start virtual services that were installed with the startup type of "Automatic". The virtual services are started when the first parent process is executed by the user. This option can be used to disable auto-starting of virtual services.

Examples

Don't start virtual services

```
[BuildOptions]
AutoStartServices=0
```

Start virtual services when first process is launched (default)

```
[BuildOptions]
AutoStartServices=1
```

Beta

See also: "AutoShutdownServices" on page 98

BlockSize

`BlockSize`—Controls this size of compression blocks used when compressing files during build

Using a larger block size can achieve higher compression, however larger block sizes can have negative impact on performance.

- The build process slows down with larger block sizes
- Startup time and file reads for applications will be slower with large block sizes
- More memory is required at runtime when larger block sizes are used

`BlockSize` can be specified in two places:

1. `Package.ini` file: in this case, the block size becomes the default for all files in the project unless otherwise specified
2. `##Attributes.ini`: in this case, the block size overrides the block size for the present directory and all subdirectories. In the manner you can use different block sizes for different directories within a single project.

Example

The default block size is 64k

```
[Compression]
BlockSize=64k
```

'other block size options follow

```
'BlockSize=128k
'BlockSize=256k
'BlockSize=512k
'BlockSize=1M
```

See also: "CompressionType" on page 100

ChildProcessEnvironme...

`ChildProcessEnvironmentExceptions`—Enable exceptions to the default child execution policy

See "ChildProcessEnvironme..." on page 99 for usage and example.

ChildProcessEnvironme...

`ChildProcessEnvironmentDefault`—Determines if child processes are run in the virtual environment by default

VMware Project North Star (Thinstall)'s default behavior is to create all child processes inside the virtual environment; this option allows you to change VMware Project North Star (Thinstall)'s behavior so that new child processes are created outside of the virtual environment.

If there are specific processes that don't match the default, you can use `ChildProcessEnvironmentExceptions` ("ChildProcessEnvironme..." on page 99) to change the execution behavior for subsets of applications.

You can also use the Script API function `ExecuteExternalProcess` ("ExecuteExternalProcess" on page 123) a nd `ExecuteVirtualProcess` ("ExecuteVirtualProcess" on page 123) to execute applications inside or outside of the virtual environment.

Examples

Specify that default behavior is to create child processes as virtual (default)

Beta

```
[BuildOptions]
ChildProcessEnvironmentDefault=Virtual
ChildProcessEnvironmentExceptions=AcroRd.exe;notepad.exe
```

Specify that default behavior is to create child processes outside of the virtual environment

```
[BuildOptions]
ChildProcessEnvironmentDefault=External
```

CommandLine

`CommandLine`—Specify command line arguments for a shortcut executable

For shortcut executables, you can specify the command line which is passed to the virtual app. You should include the app name as the first parameter.

Examples

Pass "/SomeOption SomeParameter" as command line arguments

```
[MyShortcutApp.exe]
Source=%ProgramFilesDir%\Myapp\MyShortcutApp.exe
Shortcut=HostApp.exe
CommandLine="%ProgramFilesDir%\Myapp\MyShortcutApp.exe" /SomeOption
SomeParameter
```

CompressionType

`CompressionType`—Controls what type of compression is used

VMware Project North Star (Thinstall) supports 3 different compression models: `None`, `Fast`, and `Small`

`None`—None is the default when capturing a package. This option is useful for building your application quickly for testing purposes. No compression also improves application startup time on older PCs, or when the application is launched multiple times. On subsequent executions of the same application, the Windows disk cache can provide data faster without compression enabled.

`Fast`—This option is recommend for most packages. Fast compression has a very fast rate of decompression, so it has very little impact on application startup time; it also has very little impact on memory consumption at runtime. Fast compression achieves similar compression ratios as the ZIP algorithm.

`Small`—This option is recommend when startup time is less important than package size. Small compression can achieve compression ratios slightly better than BZIP2.

Sample compression ratios and startup times for Office 2003 package running from a local hard drive:

	None	Fast	Small
Size	448,616k	257,373k	228,235k
Compression ratio	100%	57%	50%
Startup time (1st execution)	6 sec	6 sec	19 sec
Startup time (2nd execution)	0.1 sec	1 sec	17 sec
Build Time (1st build)	3 mins	19 mins	13 mins
Build Time (2nd build)	2 mins	1.2 mins	1.1 mins

CompressionType can be specified in two places:

1 `Package.ini` file: in this case, the compression type becomes the default for all files in the project unless otherwise specified

2 `##Attributes.ini`: in this case, the compression type overrides the compression algorithm for the present directory and all subdirectories. In the manner you can use different compression algorithms for different directories within a single project.

Beta

Examples

No Compression for fastest build time and fastest load time (default)

```
[Compression]
CompressionType=None
```

Fast Compression for slow build time, good compression ratio, and fast load time

```
[Compression]
CompressionType=Fast
```

Small Compression for medium build time, great compression ratio, but slow load time

```
[Compression]
CompressionType=Small
```

See Also: "BlockSize" on page 99

DirectoryIsolationMode

`DirectoryIsolationMode`—Controls default isolation mode for directories in package

`DirectoryIsolationMode` controls the default isolation mode for the package. This setting will apply to any directories that do not have an explicitly specified setting.

For example, consider a package that looks like the following:

```
ThinstallProject
               Package.ini
               %ProgramFilesDir%\MyApp\##Attributes.ini
```

Here `Package.ini` sets the default isolation mode for the project, but individual `##Attributes.ini` may change the isolation mode for specific directories and their children. Any directories that have not been specified, such as `C:\myfolder` will inherit the isolation mode from the `package.ini` file. Directories that are created under `C:\Program Files\myapp` will inherit the isolation mode from the `##Attributes.ini` file.

Examples

WriteCopy isolation allows the application to read from the host PC but not write to it (default)

```
[Isolation]
DirectoryIsolationMode=WriteCopy
```

Merged isolation allows the application to write to any location on the PC, except where the package specifies otherwise

```
[Isolation]
DirectoryIsolationMode=Merged
```

DisableTracing

`DisableTracing`—Prevents `.trace` file generation when Log Monitor is running

VMware Project North Star (Thinstall)'s Log Monitor utility ("Log Monitor" on page 74) can be used to produce a `.trace` file for applications run by VMware Project North Star (Thinstall). This option can be used to disable the ability to produce trace files for specific applications.

Examples

This prevents an application from creating a `.trace` file even if Log Monitor is running

```
[BuildOptions]
DisableTracing=1
```

This allows Log Monitor to create a `.trace` file (default)

```
[BuildOptions]
DisableTracing=0
```

Beta

It may be desirable to turn off `.trace` file generation capabilities for several reasons:

- For security purposes, it may be preferable to hide execution history from prying eyes
- During testing, it may be useful to turn off tracing for specific Thinstalled applications that are known to work, since producing extra .trace files represents a waste of disk space and CPU time

Disabled

`Disabled`—Indicates that a build target is a placeholder and thus should not generate an EXE

Example

This disables generation of the `app.exe` file during build; by changing Disabled to 0 or removing the line, one can enable generation of app.exe

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Disabled=1
```

ExcludePattern

`ExcludePattern`—Allows you to exclude specify files to exclude during the build process.

This option can be used to exclude specific files or directories from a project determined dynamically at build time. The list of patterns is specified with a comma separator where * matches 0 or more following characters and '?' matches exactly one character similar to the DOS `dir` command. Unlike DOS `dir` command syntax, wild characters can be applied to both directories names and filenames.

`ExcludePattern` can be specified in two places:

1 `Package.ini` file: in this case, the pattern exclusion will apply to the entire directory structure

2 `##Attributes.ini`: in this case, the pattern exclusion will be added to the current list of exclusions and apply only to this directory and sub directories. In this manner you can have different exclusion list for different directories in your project.

Example

```
[FileList]
```

Exclude any path that ends with .bak or .msi

```
ExcludePattern=*.bak,*.msi
```

Exclude any directories called .svn or .cvs (and all subdirectories)

NOTE because of starting "\" character this pattern will not match filenames or directories that contain ".svn" or ".cvs" in the middle of their string

---

```
ExcludePattern=\.svn,\.cvs
```

ExternalCOMObjects

`ExternalCOMObjects`—Controls whether a specific COM object CLSID will be created by VMware Project North Star (Thinstall) or by Windows.

By default, VMware Project North Star (Thinstall) creates all COM objects inside of the virtual environment instead of Windows. COM supports out-of-process (EXE) servers as well as service-based COM objects. If an application can create such COM objects on the host PC and cause these COM objects to modify the host PC, then the integrity of the host PC cannot be assured. However, if VMware Project North Star (Thinstall) executes out-of-process and services-based COM objects inside of the virtual environment, all changes made by the COM objects will be stored in the sandbox.

Beta

Example usage:

This instructs VMware Project North Star (Thinstall) to execute 2 COM objects outside of the virtual environment if they are created by the application

```
[BuildOptions]
ExternalCOMObjects={8BC3F05E-D86B-11D0-A075-00C04FB68820};{7D096C5F-AC08-4F1F-BEB7-5C22C517CE39}
```

ExternalDLLs

`ExternalDLLs`—Force some DLLs to be loaded by Windows.

By default, VMware Project North Star (Thinstall) determines whether it should load DLLs itself or pass the loading on to Windows. If the DLL is located in the virtual filesystem, VMware Project North Star (Thinstall) will load the DLL itself. In some circumstances, it is required to have Windows load the DLL, even if it is in the virtual filesystem. An example of this is a DLL that is "injected" in other processes, using a mechanism known as "Windows hooks". For hooks to work, the DLL implementing the hook must be available on the host filesystem and be loaded by Windows. When you specify a DLL in ExternalDLLs, the DLL is extracted from the virtual filesystem into the sandbox and Windows is instructed to load it from there.

Note that the usefulness of this option is limited. If the DLL depends on other DLLs which are located in the virtual filesystem Windows won't be able to load it. You can check dependencies using Dependency walker (http://www.dependencywalker.com/)

Example:

This instructs VMware Project North Star (Thinstall) to pass loading of "inject.dll" and "injectme2.dll" on to Windows

```
[BuildOptions]
ExternalDLLs=inject.dll;injectme2.dll
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.324 or higher

Icon

`Icon`—Indicates which icon file to use for the generated `EXE` file

By default each generated application will use the main group icon from its source EXE and the individual icon resource pointed to by the group icon. You can use an alternate icon by specifying an `.ico` file or `.exe` file.

Examples

Specify `NULL` to generate an EXE with no icons

NOTE `NULL` cannot be used if the FileTypes directive is used, because one icon per filetype is allocated in the EXE image

---

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=NULL
```

Specify application icon using a EXE different from the Source EXE

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myapp\app2.exe
```

You can optionally specify which set to use by appending ",1" ",2" to the end of the Icon path name like this:

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myapp\app2.exe,1
```

Beta

Specify application icon using a `.ico` file

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myap\myicon.ico
```

See also: "RetainAllIcons" on page 111

NOTE The Windows Shell has limitations regarding display of icons for large EXE files, an easy work-around can be found here. ("Large Packages" on page 155)

InventoryName

`InventoryName`—Optional string that is used for package indentification by inventory control systems

Example usage:

`InventoryName` is typically a version independent string that tracks a specific resource of interest to inventory scanning applications

```
[BuildOptions]
InventoryName=Microsoft Office 2003
```

`InventoryName` is meant to be a version-independent string that can be used to track a VMware Project North Star (Thinstall) package with inventory-scanning software.

When deploying new versions of an application, it may be desirable to change the `SandboxName` so that the new version will have isolated user settings; however, `InventoryName` can be left constant across versions of the same application.

`InventoryName` is not used by VMware Project North Star (Thinstall) during build or execution.

IsolatedMemoryObjects

`IsolatedMemoryObjects`—List specific shared memory objects to isolate from other applications

Shared memory objects are created by applications using `CreateFileMapping` and `OpenFileMapping`. Shared memory objects can be named or anonymous; when the objects are named, they will be visible to other applications running in the same user account. Sometimes, it is desirable to isolate shared memory objects so that virtual applications cannot see system objects and vice-versa. By default, VMware Project North Star (Thinstall) will only isolate shared memory objects used by embedded Internet Explorer instances, because there is a known conflict with between explorer.exe and iexplore.exe when they map sandboxed files. You can use this option to isolate additional named shared memory objects so they are visible only to other virtual applications using the same sandbox.

`IsoaltedMemoryObjects` accepts a list of entries that are separated using the ';character. Each entry can have wildcard characters '*and '?to match variable patterns.

Example

Isolate two shared memory objects, matching anything with "outlook" in the name and one matching exactly "My Shared Object"

```
[BuildOptions]
IsolatedMemoryObjects=*outlook*;My Shared Object
```

See Also: "IsolatedSynchronizatio..." on page 105

This option requires VMware Project North Star (Thinstall) 3.133 or higher.

Beta

IsolatedSynchronizatio...

`IsolatedSynchronizationObjects`—List specific synchronization objects to isolation from other applications

Windows has several different named Synchronization objects:

- `Mutex`, accessed using OpenMutex & CreateMutex
- `Semaphore`, accessed using OpenSemaphore & CreateSemaphore
- `Events`, accessed using OpenEvent & CreateEvent

By default VMware Project North Star (Thinstall) will not isolate Synchronization objects. You can specify a list of synchronization objects to isolate from other applications not running in the same virtual namespace. A namespace is defined by the location of the application's sandbox. If two applications share the same sandbox path, they will have the same namespace for isolated SynchronizationObjects. If two applications have the same sandbox name, but the path to the sandbox is different, the applications will have separate namespaces.

`IsolatedSychronizationObjects` accepts a list of entries that are separated using the ';character. Each entry can have wildcards characters '*and '?to match variable patterns.

Example

Isolate two synchronization objects, matching anything with "outlook" in the name and one matching exactly "My Shared Object"

```
[BuildOptions]
IsolatedSychronizationObjects=*outlook*;My Shared Object
```

See also: "IsolatedMemoryObjects" on page 104

This option requires VMware Project North Star (Thinstall) 3.133 or higher.

LogPath

`LogPath`—Controls where to store .trace files when logging

Example

This directs VMware Project North Star (Thinstall) to store log files in `c:\ThinstallLogs`

NOTE Unlike most paths in VMware Project North Star (Thinstall), LogPath cannot contain macros such as %AppData% or %Temp%

```
[BuildOptions]
LogPath=C:\ThinstallLogs
```

MSIDefaultInstallAllUsers

`MSIDefaultInstallAllUsers` – Set default installation mode of the MSI database

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106).

When set to "1", the default installation mode of the generated MSI database is set to "per-machine". If you want the default mode to be "per-user", set this option to "0". Setting it to "2" will result in a default mode of per-machine for Administrators and a per-user for non-Administrators.

Example

This directs VMware Project North Star (Thinstall) to generate an MSI which is installed on a per-user bases by default

Beta

NOTE by suppling command-line arguments to msiexec you can force a per-machine install

For example `msexec.exe mymsi.msi ALLUSERS=1`

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=0
```

This directs VMware Project North Star (Thinstall) to generate an MSI which is installed on a per-machined bases by default

NOTE by suppling command-line arguments to msiexec you can force a per user-machine install

For example `msexec.exe mymsi.msi ALLUSERS=`

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=1
```

This directs VMware Project North Star (Thinstall) to generate an MSI which is installed on a per-machined for Administrators and per-user for non-Administrators if the user does not have permission to install on a per-machine bases,

NOTE by suppling command-line arguments to msiexec you can force a per user-machine install

For example `msexec.exe mymsi.msi ALLUSERS=`

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=2Availability:
```

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

Beta

MSIFilename

`MSIFilename` – Enable generation of an MSI database and specify its filename

If set, this option causes the build to produce a Windows Installer with the specified filename in the output directory. For more information, see Chapter 7, "MSI Generation," on page 93.

Example

This directs VMware Project North Star (Thinstall) to generate an MSI during build, replace mymsi.msi with your own filename

```
[BuildOptions]
MSIFilename=mymsi.msi
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIInstallDirectory

`MSIInstallDirectory` – Specify the final path component of the default installation directory

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106).

By default, packages will install in `%ProgramFilesDir%\<InventoryName> (Thinstalled)` (for a per-machine install). You can change the last component in the default install path by using the MSIInstallDir option. For example, if you set `"MSIInstallDirectory=ExampleDir"` then the default install dir (for per-machine installs) will be `"%ProgramFilesDir%\ExampleDir"`.

This option can only be used to change the last component of the default install path. If you want to change the whole path, specify a `"INSTALLDIR=<full_path>"` on the installer command line.

This creates a `.msi` file that will be installed to `c:\Program Files\My Applicaiton\`

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIInstallDirectory=My Application
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIManufacturer

`MSIManufacturer` – Specify the manufacturer to put in the MSI database

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106).

Set this option to the name of your organization. It will be displayed when you show the properties of the database, but has no effect otherwise.

This creates a `.msi` file with the Manufactor set to "My Company Name"

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIManufacturer=My Company Name
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIProductCode

`MSIProductCode` – Specify a product code for the MSI database

This option has an effect only when generation of a Windows Installer database is requested via the MSIFilename option ("MSIFilename" on page 106).

Each MSI database needs a ProductCode. SetupCapture will generate a suitable default ProductCode and place it in `Package.ini`. If you change it, make sure the new value is a valid GUID (Globally Unique IDentifier).

For more information, see Chapter 7, "MSI Generation," on page 93.

This creates a `.msi` file with a specific product code

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIProductCode={590810CE-65E6-3E0B-08EF-9CCF8AE20D0E}
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIProductVersion

`MSIProductVersion` – Specify a product version number for the MSI database

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106).

The product version number will be displayed when you show the properties of the database. When deploying to a machine that already has the package installed, Windows Installer will check the version numbers and will refuse to install an older version over a newer version, you'll have to manually uninstall the old version first.

This creates a `.msi` file with a specific product version

Beta

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIProductVersion=1.0
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIRequireElevatedPri...

`MSIRequireElevatedPrivileges` – Mark the MSI database as "requires elevated privileges"

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106). Also, this option has an effect on Vista only.

When this option is set to "1", the generated MSI database will be marked as requiring elevated privileges. If your system is set up for UAC prompts, this results in a UAC prompt when installing.

When you set the option to "0", no UAC prompt will be given, but you won't be able to install machine-wide.

For more information, see Chapter 7, "MSI Generation," on page 93.

This creates a `.msi` file with always prompts for elevated privileges on Vista

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIRequireElevatedPrivileges=1
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

MSIUpgradeCode

`MSIUpgradeCode` – Specify an upgrade code for the MSI database

This option has an effect only when generation of a Windows Installer database is requested via the `MSIFilename` option ("MSIFilename" on page 106).

It is highly recommended that each MSI database has an UpgradeCode. SetupCapture will generate a suitable default UpgradeCode and place it in Package.ini. There should be no need to change it. If you do change it, make sure the new value is a valid GUID (Globally Unique IDentifier).

For more information, see Chapter 7, "MSI Generation," on page 93.

This creates a `.msi` file with a specified upgrade code

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIUpgradeCode={D89F1994-A24B-3E11-0C94-7FD1E13AB93F}
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.210 or higher

NetRelaunch

`NetRelaunch`—Controls whether to relaunch an application from the local disk when running from a net share or removable disk

By default, VMware Project North Star (Thinstall) will automatically detect if an application is being run from a network drive or removable disk, and will relaunch the application using a local fixed disk. This is to resolve a problem in which Symantec Anti-Virus tries to perform a complete scan of an EXE under some conditions, and this scan can have a big impact on launch times for large EXE files located network shares.

When Symantec AV decides to perform a full file-scan:

Beta

- If EXE is launched from a network share or removable disk (it skips the scan when the EXE is located on the hard drive)
- When the EXE makes its first network connection (it does not scan the EXE if the EXE does not make any network connections)

Because a large number of desktops have SAV installed, VMware Project North Star (Thinstall) automatically compensates for this by allowing applications to launch from a network share without incurring the lengthy scan times. It does so by creating a small stub executable in the user's sandbox which is then relaunched. Because the small executable can be scanned quickly it will load the remainder of the application data from the original source location. You can disable this default behavior by adding the package.ini option:

If your application is small in size or you know that SAV is not installed on the desktops you are deploying to, you may want to turn off NetRelaunch for better first-time launch performance.

Examples

This will disable relaunching of the application locally on a fixed disk

```
[BuildOptions]
NetRelaunch=0
```

If launched from a network drive, re-launch using a local stub (default)

```
[BuildOptions]
NetRelaunch=1
```

NoRelocation

`NoRelocation`—Strips relocation information from the resulting executable

Windows executables can optionally contain base relocation information which enables Windows to load the EXE image at an alternate starting memory address. If the base address is above 0x40000 the executable will always be loaded at its specified base address, so relocation information is not needed and can be safely stripped out of the resulting image. Relocation information is typically fairly small on disk, so removing this information will not have a big impact on the size of the EXE files generated.

Example

This will strip relocation information from the generated `EXE` file

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
NoRelocation=1
```

RegistryIsolationMode

`RegistryIsolationMode`—Controls default isolation mode for registry keys in package

`RegistryIsolationMode` controls the default isolation mode for the package. This setting will apply to any registry keys that do not have an explicitly specified setting.

Examples

WriteCopy isolation allows the application to read keys from the host PC but not write to it (default)

```
[Isolation]
RegistryIsolationMode=WriteCopy
```

Merged isolation allows the application to write to any key on the PC, except where the package specifies otherwise

```
[Isolation]
RegistryIsolationMode=Merged
```

Beta

PermittedGroups

`PermittedGroups`—Restricts usage of package to a specific set of Active Directory Groups

Example

Specify a list of AD Group names separated by ';' [BuildOptions] can set default for apps in project that can be overwritten by individual [App.exe] sections

```
[BuildOptions]
PermittedGroups=Administrator;OfficeUsers
AccessDeniedMsg=You do not have permission to execute this application, please call support @
1-800-822-2992
```

Overwrite global PermittedGroups just this specific application

```
[App1.exe]
PermittedGroups=Guest
AccessDeniedMsg=You do not have permission to execute this application, please call support @
1-800-822-2992
```

Inherit PermittedGroups from [BuildOptions]

```
[App2.exe]
...
```

While building of a package, VMware Project North Star (Thinstall) will lookup the specified Group Names into SID values so the Group Names specified must be valid at the time of build. VMware Project North Star (Thinstall) can resolve group ownership at runtime using cached creditials so laptop users can continue to be authenticated even when they are offline.

If the user does not have access to execute the package, the AccessDeniedMsg ("AccessDeniedMsg" on page 97) can be customized to instruct the user.

ReserveExtraAddress...

`ReserveExtraAddressSpace`—Indicates how much extra address space needs to be reserved for the Thinstalled executable

Normally, TLink will set the SizeOfImage field in the generated executable based on the SizeOfImage field of the source executable. The SizeOfImage field is used by the Windows loader to determine how much virtual address space to reserve for the executable. In special circumstances (when you're building a package based on a source executable which is not included in the package), you can reserve extra virtual address space by specifying the ReserveExtraAddressSpace option. The value is the number of extra bytes to reserve. Optionally, you can follow the number by "K" to indicate kilobytes or "M" to indicate megabytes. The default value is 0 (don't reserve extra address space.

Examples

Tell the Windows loader to reserve 512KB of extra address space

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
ReserveExtraAddressSpace=512K
```

Tell the Windows loader to not reserve any extra address space (default)

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
ReserveExtraAddressSpace=0
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.215 or higher

Beta

RetainAllIcons

`RetainAllIcons`—Indicates all of the Source EXEsoriginal icons should be retained in the generated Thinstalled EXE file

By default TLink will construct a new EXE using a source executable. To reduce disk space, the new EXE image will contain only icons viewable from the system shell, while all other icons are stored inside of the package and are still accessible to the application while it is running. The icons that are accessible by the system cannot be compressed, so their disk size is larger. In some cases it may be desirable to have all of the application's original icons visible to the system shell.

Examples

Instruct Tlink to retain all of the application's original icons

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
RetainAllIcons=1
```

Instruct Tlink to strip out unused icons from the system visible portion of the EXE file (default)

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
RetainAllIcons=0
```

See also: Icon ("Icon" on page 103)

RemoveSandboxOnExit

`RemoveSandboxOnExit`—Resets the application by deleting the sandbox when the last child process exits

All application modifications to registry and filesystem locations that have isolation modes set to WriteCopy or Full will go to a `"Sandbox"` directory. By default, the sandbox directory is left behind so that settings will be persistent across multiple executions of the application. In some cases, it may be desirable to "wipe the slate clean" each time the application exits. This option instructs VMware Project North Star (Thinstall) to delete the sandbox directory when the application exits. If the application creates child processes, the sandbox will not be deleted until all child processes exit. In some cases, applications may leave children behind by design; this can stop the clean-up operation from occurring. For example Office 2003 leaves behind a process called ctfmon.exe. It may be necessary to use a script to kill ctfmon.exe and similar children in order to force this cleanup operation to occur.

NOTE You can also decide dynamically at runtime whether to delete the sandbox on exit by using the Script API function `"RemoveSandboxOnExit"` ("RemoveSandboxOnExit" on page 127)

Examples:

This instructs VMware Project North Star (Thinstall) to delete the sandbox when the application exits

```
[BuildOptions]
RemoveSandboxOnExit=1
```

This instructs VMware Project North Star (Thinstall) to leave sandbox behind when the application exits (default)

```
[BuildOptions]
RemoveSandboxOnExit=0
```

SandboxName

`SandboxName`—Sets the name of directory where the sandbox is created and stored

Example:

```
[BuildOptions]
SandboxName=My Application 1.0
```

Beta

The SandboxName is used when creating a new Sandbox; for the example above, the default sandbox path would be:

```
C:\Documents and Settings\USERNAME\Application Data\Thinstall\My Application 1.0
```

When upgrading an application, the Sandbox name plays an important role in determining if the user will retail their previous personal settings or use new settings. By changing the Sandbox Name with new deployments, one can decide whether the user will create a new Sandbox with different settings or retain the same Sandbox.

SandboxPath

SandboxPath—Controls the path where VMware Project North Star (Thinstall) will create a new sandbox by default

Examples:

This creates the default Sandbox in the same directory as the EXE

```
[BuildOptions]
SandboxPath=.
```

This creates the default Sandbox in a subdirectory under where the EXE is located

```
[BuildOptions]
SandboxPath=LocalSandbox\Subdir1
```

This creates the default Sandbox in the user's AppData folder under the subdirectory Thinstall

```
[BuildOptions]
SandboxPath=%AppData%\Thinstall
```

This creates the default Sandbox on a network mapped drive

```
[BuildOptions]
SandboxPath=Z:\Sandboxes
```

If an application is meant to run only from portable media such as USB Flash devices, SandboxPath= can be used to force the application to use a local sandbox. It is also possible to control the default location of the Sandbox using environment variables or by creating a Thinstall directory. Environment variables and a local Thinstall directory will take precedence over the path specified by SandboxPath.

See Chapter 5, "Sandbox Overview," on page 65 for more details on Sandbox creation and probing works.

This option requires VMware Project North Star (Thinstall) 3.133 or higher.

SandboxNetworkDrives

SandboxNetworkDrives—Determines if network-mapped drives will have sandboxing applied.

By default VMware Project North Star (Thinstall) allows users to write directly to network-mapped drives without applying sandboxing.

Examples:

This will prevent the user from writing directly to network-mapped drives; instead, changes go to the sandbox

```
[BuildOptions]
SandboxNetworkDrives=1
```

This allows users to write directly to network-mapped drives without changes going to the sandbox

```
(default)
[BuildOptions]
S9andboxNetworkDrives=0
```

SandboxRemovableDisk

SandboxRemovable—Determines if removable drives will have sandboxing applied.

Beta

By default VMware Project North Star (Thinstall) allows users to write directly to removable drives without applying sandboxing. Removable disks include USB Flash and removable hard drives.

Examples:

This will prevent the user from writing directly to removable disk; instead, changes go to the sandbox

```
[BuildOptions]
SandboxRemovableDisk=1
```

This allows users to write directly to removable disks without changes going to the sandbox (default)

```
[BuildOptions]
SandboxRemovableDisk=0
```

StripVersionInfo

`StripVersionInfo`—Removes all version information from the source EXE when building the target application

Version information in an executable can be found in the "Properties" dialog from the windows shell; typically, this includes Copyright, Trademark, and Version number information. By default, VMware Project North Star (Thinstall) will copy all version information from the source EXE (specified using Source) ("Source" on page 113). This option can be used to strip version information from the resulting application.

Example usage:

This will generate a target application with no version information

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
StripVersionInfo=1
```

See also: "Version.XXXX" on page 115

Source

`Source`—Points to the EXE file which will be initially loaded by VMware Project North Star (Thinstall)

`Source` is specified on a per-executable basis. If an application suite has 3 accessible user entry points; for example, `Winword.exe`, `Powerpnt.exe`, and `Excel.exe`, your `package.ini` will list 3 application entries, and each entry will have a different Source entry.

Examples

Create an user-accessible entry point for `C:\Program Files\Myapp\app1.exe`

```
[app1.exe]
Source=%ProgramFilesDir%\Myapp\app1.exe
```

Create an user-accessible entry point for `C:\Program Files\Myapp\app2.exe`

```
[app2.exe]
Source=%ProgramFilesDir%\Myapp\app2.exe
```

Shortcut

`Shortcut`—Points to the name of the VMware Project North Star (Thinstall)-generated EXE which will host the package data.

Example

Create a "shortcut" application that references `HostApp.exe`

`HostApp.exe` should be listed somewhere in the package and also needs to be present in the same directory in order for `MyShortcutApp.exe` to run

```
[MyShortcutApp.exe]
```

Beta

```
Source=%ProgramFilesDir%\Myapp\MyShortcutApp.exe
Shortcut=HostApp.exe
```

UpgradePath

`UpgradePath`—Location to probe for application updates

By default, VMware Project North Star (Thinstall) will look for application upgrades in the same directory as the main executable file. This option can be used to specify an alternate location to look for application upgrades.

Example

This instructs VMware Project North Star (Thinstall) to probe for application upgrades under `C:\Program Files\MyAppUpgrades`

```
[BuildOptions]
UpgradePath=%ProgramFilesDir%\MyAppUpgrades
```

See also: How upgrades work (“Upgrading applications” on page 57)

VirtualizeExternalOut...

`VirtualizeExternalOutOfProcessCOM`—Controls whether external Out-of-process COM objects are run in the virtual environment

Thinstalled applications can create COM objects that are registered in the virtual environment, as well as COM objects from the host system.

This option determines how to treat Out-of-process COM objects that are not part of a VMware Project North Star (Thinstall) package and not registered in the virtual registry. By default, VMware Project North Star (Thinstall) will execute external out-of-process COM objects in the virtual environment, so such COM objects cannot modify the host PC. If you run into a compatibility issue with an external COM object running in the virtual environment, this option can be used to allow such objects to be created by and run on the host system. If you want to run only specific COM objects outside of the virtual environment, you can list each COM object's CLSID explicitly using ExternalCOMObjects (“ExternalCOMObjects” on page 102).

Examples:

This instructs VMware Project North Star (Thinstall) to execute all external out-of-process COM objects in the system context, not in the virtual environment

```
[BuildOptions]
VirtualizeExternalOutOfProcessCOM=0
```

This instructs VMware Project North Star (Thinstall) to execute all external out-of-process COM objects in the virtual environment (default)

```
[BuildOptions]
VirtualizeExternalOutOfProcessCOM=1
```

VirtualDrives

`VirtualDrives`—Specifies additional drive letters that should be available to the application at runtime

Virtual drives can help solve issues in which applications rely on hard-coded paths to specific drive letters which may or may not be available on the client PCs you are distributing to. For example, some legacy applications may be designed to expect that the D drive is a CDROM and that various data files will be available at `D:\media`.

Virtual drives are visible only to applications running in the virtual environment; they do not have any impact on the real Windows environment. Isolation modes (Chapter 4, “Isolation Modes,” on page 61) for virtual drives will be inherited from the project's default isolation mode unless you specific override it, as shown below. If a virtual drive is setup with IsolationMode=Merged, any writes to that drive will fail if it does not exist on the real system.

Beta

Projects created by Setup Capture ("Setup Capture" on page 69) will automatically list virtual drive information for drives that were present at the time of capture.

VirtualDrives is specified as a single string which can hold information for multiple drive letters and optional parameters for those drive letters. The format for VirtualDrives should look like this:

```
VirtualDriv[es= Drive=A, Serial=12345678, Type=REMOVABLE; Drive=B, Serial=9ABCDEF0, Type=FIXED
```

The ';' is used to separate drive letters, and ',' is used to separate parameters for individual drive letters.

`Drive=`(single character between 'a' and 'z')

`Serial=`(8-digit hex number)

`Type=`

`FIXED`—The drive has fixed media; for example, a hard drive or internal flash drive.

`REMOVABLE`—The drive has removable media; for example, a floppy drive, thumb drive, or flash card reader.

`CDROM`—The drive is a CD-ROM drive.

`RAMDISK`—The drive is a RAM disk.

Examples:

The simplest usage is to specify a single virtual Drive letter; by default, it will be assigned a serial number and type of FIXED

```
[Build]
lDrives=Drive=M
```

This specifies 3 virtual drive letters (X, D, and Z):

Drive X will appear to be a Removable disk with Serial number ff797828

Drive D will appear to be a CD-ROM drive with an automatically assigned Serial number

Drive Z will appear to be a FIXED disk with an automatically assigned Serial number

---

NOTE the ';' character is used to separate information assigned to different drive letters

---

```
[BuildOptions]
VirtualDrives=Drive=X, Serial=ff897828, Type=REMOVABLE; Drive=D, Type=CDROM; Drive=Z
```

How to change the isolation mode for a Virtual Drive

1 Add the folder `%Drive_X%` to your VMware Project North Star (Thinstall) project

2 In the new directory, place an `##Attributes.ini` file ("##Attributes.ini" on page 150) to specify the isolation mode for this drive letter.

This option requires VMware Project North Star (Thinstall) version 3.146 or higher.

Version.XXXX

`Version.XXXX`—Used to override EXE version strings or add new version strings

Version resources are normally copied from the original .exe. You can override the version resource strings (and add new ones) using a "Version.<string_name>=string_value" setting.

Example

Sets the VersionInfo field "ProductName" to the value "My New Product Name"

```
[Application.exe]
Version.ProductName=My New Product Name
Version.Description=This Product is great!
```

This option requires VMware Project North Star (Thinstall) 3.133 or higher

Beta

WorkingDirectory

`WorkingDirectory`—Sets the current working directory before the application starts

This option can set the CWD (Current Working Directory) for individual applications. The CWD setting is applied before the application starts. The CWD value does not need to exist on the system.

Example

Set's the current working directory to `C:\Program Files\My Application`

```
[Application.exe]
WorkingDirectory=%ProgramFilesDir%\My Application
```

Beta

Scripting

9

Scripting allows you to execute custom code before starting an application packaged with VMware Project North Star (Thinstall) or after an application exits.

Callback functions ("Callback functions" on page 117) allow you to specify when blocks of code will execute.

API functions (Chapter 10, "API functions," on page 121) allow you to execute VMware Project North Star (Thinstall)-specific functions and interact with the VMware Project North Star (Thinstall) runtime.

Q. How do I add scripts to my application?

A. Create a ansi text file with the `.vbs` file extension in the root project directory for an application (i.e. the same directory where `package.ini` is located). During the build process, VMware Project North Star (Thinstall) will add all of the script files to your EXE file and then at runtime, it will execute each of these script files.

Q. My script is executing too many times, how can I execute it only once?

A. Many applications will create child processes, and scripts will execute for child process as well. To execute code only in the main parent process, use callback functions ("Callback functions" on page 117).

Q. What is the format for script files, and what functions can I use?

VMware Project North Star (Thinstall) uses VB Script to execute script files, so any valid VB Script will load an execute under VMware Project North Star (Thinstall). Click here for Microsoft's VB Script User Guide (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/dd5dc02a-71e4-412b-8b30-9cc2d3d5e6fb.asp). VB script can be used to access COM controls registered on the host system, or registered within the virtual package.

Callback functions

Callback functions with specific names will execute only under certain conditions. For example, callback functions allow script code to execute only when an application starts or quits.

Currently defined callback function names:

`OnFirstSandboxOwner`—Called only when an application first locks the sandbox. This callback will not be called if a 2nd copy of the same application is launched using the same sandbox which the first copy is still running. If a 1st application spawns sub-process and then quits, the sandbox remains locked by the 2nd sub-process so this callback will not execute again until all sub processes have quit and the application is run again.

`OnFirstParentStart`—Called prior to executing a VMware Project North Star (Thinstall) EXE file regardless of weither the sandbox is simutaneously owned by another Thinstalled EXE.

`OnFirstParentExit`—Called when the first parent process exits. If a parent process executes a child process and quits, this callback will be called even if the child process continues to execute.

Beta

`OnLastProcessExit`—Called when the last process owning the sandbox exits. If a parent process executes a child process and quits, this callback will be called when the last child process exits.

```
------------------------example.vbs ---------------------------------
Function OnFirstSandboxOwner
                msgbox "The sandbox owner is: " + GetCurrentProcessName
End Function

Function OnFirstParentExit
                msgbox "Quiting application: " + GetCurrentProcessName
End Function

msgbox "This code will execute for all parent and child processes"
-----------------------------------------------------------------------
```

Examples

Timeout an application on a specific date ("timeout example" on page 118)

Run a `.bat` file from a network share inside the virtual environment (".bat example" on page 118)

Modifying the Virtual Registry ("Registry Modify" on page 119)

Import `.reg` file at runtime (".reg example" on page 119)

Stopping a virtual service when the main application quits ("Stopping Service" on page 119)

Copying an external system configuration file into the virtual environment on startup. ("copyfile example" on page 119)

How to use an example:

1 Save contents to a plain text file with the `.vbs` extension in the same directory as your `package.ini` file (produced by SetupCapture).

> **NOTE** It doesn't matter what filename you use, all `.vbs` files will be added to the package at build time and run

2 Rebuild the application

.bat example

- This script will execute an external `.bat` file from a network share inside of the virtual environment
- The `.bat` file can make modifications to the virtual environment by copying files, deleting files, or apply registry changes using `regedit /s regfile.reg`
- We need to make sure we only execute this for the first parent process otherwise each copy of `cmd.exe` will execute the script and we'll have infinite recursion

```
Function OnFirstParentStart
                Set Shell = CreateObject("Wscript.Shell")
                Shell.Run "\\jcdesk2\test\test.bat"
End Function
```

timeout example

This script will prevent an application from being used after the date March 20, 2007

> **NOTE** This check is performed on the parent process startup as well as any child process startup

```
ExpirationDate = CDate("03-20-07")

if Date >= ExpirationDate then
                msgbox "This application has expired, please contact Administrator"
                ExitProcess 0
```

Beta

```
end if
```

Registry Modify

This example modifies the virtual registry at runtime so that an external ODBC driver can be loaded from the same directory where the package EXE is located.

Get path to package EXE files

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

Find last slash in path and grab just the characters before this

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Form a new path to the ODBC DLL file located outside of the package

```
DriverPath=SourcePath + "tsodbc32.dll"
```

Now modify the virtual registry so that it points to this location, this will cause the application to load the DLL from an external location

```
Set WSHShell = CreateObject("Wscript.Shell")
WSHShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\Transoft ODBC Driver\Driver",
DriverPath
```

.reg example

This script will import the registry values from an external `.reg` file into the virtual registry at runtime

```
Function OnFirstParentStart
               ExecuteVirtualProcess "regedit /s c:\tmp\somereg.reg"
               End Function
```

Stopping Service

This script will stop a virtual or real service when the “main” application quits

```
Function OnFirstParentExit
               Set WshShell = CreateObject("WScript.Shell")
               WshShell.Run "net stop ""iPod Service"""
End Function
```

copyfile example

The following scripts demonstrates how you can copy a configuration file located in the same directory as your Thinstalled EXE into the virtual filesystem each time the application starts one use for this is to allow you to have an external configuration file which is easy to edit after deployment because the copy occurs each time you run the application, any edits to the external version will be reflected in the virtual version

For example if your Thinstalled EXE is running from `\\server\share\myapp.exe`, this script will look for a config file located at `\\server\share\config.ini` and copy it to the virtual filesystem location: `c:\program files\my application\config.ini`

By putting this code in the OnFirstParentStart function, it will only be called once per execution Otherwise it will be executed for every child process as well

```
Function OnFirstParentStart
```

`TS_ORIGIN` is set by VMware Project North Star (Thinstall) to indicate the full path to the Thinstalled EXE package

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

We want to chop off the filename from TS_ORIGIN, so find the last backslash and remove everything after this

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Beta

The source file to copy into the virtual environment is the package path plus `config.ini`

```
SourceFile = SourcePath + "Config.ini"
```

The location we want to copy to may be in a different location on different computers if the `Program Files` directory is mapped to a different location than `c:\` The following call will let VMware Project North Star (Thinstall) expand a macro to get the correct location for the local PC

```
DestFile = ExpandPath("%ProgramFilesDir%\MyApplication\Config.ini")
```

Use the FileSystemObject to check to make sure the Source file exist

```
Set objFSO = CreateObject("Scripting.FileSystemObject")
If objFSO.FileExists(SourceFile) Then
```

If the source file exists, copy it into the virtual filesystem we are assuming the virtual directory `%ProgramFilesDir%\MyApplication` already exists in the package

```
        objFSO.CopyFile SourceFile, DestFile, TRUE
        End if
        End Function
```

System registry example

This script will add a value to the system registry.

We do this by creating a `.reg` file and then executing "`regedit /s`" as an external process so it will access the system registry instead of the virtual registry.

```
Function OnFirstParentStart
```

We need to create the `.reg` file in a place which has IsolationMode set to Merged, so we can access it easily from both the virtual environment (this script) and the system environment (regedit /s). One directory which has a Merged IsolationMode by default is the `%Personal%` directory, so let's create the reg file there

```
RegFileName = ExpandPath("%Personal%\thin.reg") ("ExpandPath" on page 122)
Set fso = CreateObject("Scripting.FileSystemObject")
Set RegFile = fso.CreateTextFile(RegFileName, true)
```

Construct the `.reg` file

```
RegFile.WriteLine("Windows Registry Editor Version 5.00")
RegFile.WriteBlankLines(1)
RegFile.WriteLine("[HKEY_CURRENT_USER\Software\Thinstall\demo]") RegFile.WriteLine(chr(34) &
"InventoryName" & chr(34) & "=" & chr(34) & GetBuildOption("InventoryName") & chr(34))
("GetBuildOption" on page 124)
RegFile.Close
```

Enter the info into the system registry, wait until it is done

```
RegEditPid = ExecuteExternalProcess("regedit /s " & chr(34) & RegFileName & chr(34))
("ExecuteExternalProcess" on page 123)
WaitForProcess RegEditPid, 0 ("WaitForProcess" on page 128)
```

Clean up

```
        fso.DeleteFile(RegFileName)
End Function
```

Beta

API functions 10

`AddForcedVirtualLoadPath`—Forced DLLs located outside the package to be loaded as virtual DLLs ("AddForcedVirtualLoadPath" on page 121)

`ExitProcess`—Immediately quits the current process ("ExitProcess" on page 122)

`ExpandPath`—Expands a path which is in Macro format ("ExpandPath" on page 122)

`ExecuteExternalProcess`—Executes a command on the system outside of the virtual environment ("ExecuteExternalProcess" on page 123)

`ExecuteVirtualProcess`—Executes a command inside of the virtual environment ("ExecuteVirtualProcess" on page 123)

`GetFileVersionValue`—Returns version information from a DLL or EXE ("GetFileVersionValue" on page 124)

`GetCommandLine`—Returns the full commandline and parameters passed to the current process ("GetCommandLine" on page 125)

`GetCurrentProcessName`—Returns the full path name of the current process inside the virtual environment ("GetCurrentProcessName" on page 125)

`GetOSVersion`—Returns a description of the current Windows OS version ("GetOSVersion" on page 125)

`GetEnvironmentVariable`—Retrieves a system environment variable ("GetEnvironmentVariable" on page 126)

`RemoveSandboxOnExit`—Tells VMware Project North Star (Thinstall) to delete the sandbox directory when the last process exits ("RemoveSandboxOnExit" on page 127)

`SetEnvironmentVariable`—Sets an environment variable for the current process, changes are inherited to child processes. ("SetEnvironmentVariable" on page 127)

`SetFileSystemIsolation`—Sets the isolation mode of a directory ("SetFileSystemIsolation" on page 127)

`SetRegistryIsolation`—Sets the isolation mode of a registry key ("SetRegistryIsolation" on page 128)

`WaitForProcess`—Waits until a specified process has completed exeuction ("WaitForProcess" on page 128)

AddForcedVirtualLoadPath

Function AddForcedVirtualLoadPath(Path)

This function instructs VMware Project North Star (Thinstall) to load all DLLs from the specified path as virtual DLLs even if they are not located in the package. This should be used if the application needs to load external DLLs that have dependencies on DLLs located inside the package.

Beta

Parameters:

Path

[in] The filename or path for DLLs to load as virtual

Example:

This sample script will load any DLL located in the same directory as the EXE as a virtual DLL

TS_ORIGIN is the path where the EXE is running from

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

We want to chop off the filename from TS_ORIGIN, so find the last backslash and remove everything after this

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Tell VMware Project North Star (Thinstall) to load all DLLs in the same directory (or deeper) from where the source EXE is

This allows us to drop additional files in the SourcePath tree and have them resolve imports against virtual DLLs

```
AddForcedVirtualLoadPath(SourcePath)
```

ExitProcess

Sub ExitProcess(ExitCode)

This function quits the current process and sets the specified Error Code.

Parameters:

ExitCode

[in] The error code to set. This information may be available to a parent process. Typically a value of 0 is used to indicate no error.

Example:

Exit the process and indicate success

```
ExitProcess 0
```

NOTE As the process exits, the scripting system will receive its `Function OnEndProcess` callback. Also any DLLs loaded will run their termination code so they can properly clean up.

ExpandPath

Function ExpandPath(InputPath)

This function converts a path from macro format to system format.

Parameters:

InputPath

[in] A path in macro format

Returns:

The expanded macro path in system format

Beta

Example:

```
Path = ExpandPath("%ProgramFilesDir%\Myapp.exe")
```

Path = `c:\program files\myapp.exe`

NOTE All macro paths must escape the '%' and '#' characters by replacing these characters with #25 and #23

ExecuteExternalProcess

Function ExecuteExternalProcess(CommandLine)

This function executes a command outside of the virtual environment. It can be used if you need to make real system changes.

Parameters:

CommandLine

[in] CommandLine represents the application and commandline parameters to execute outside of the virtual environment

Returns:

Integer process ID, the process ID can be used with WaitForProcess ("WaitForProcess" on page 128)

Example:

```
ExecuteExternalProcess("cmd.exe /c copy c:\systemfile.txt c:\newsystemfile.txt")
```

Execute a command which requires quotes in the command-line

```
ExecuteExternalProcess("regsvr32 /s " & chr(34) & "c:\program files\my.ocx" & chr(34))
```

ExecuteVirtualProcess

Function ExecuteVirtualProcess(CommandLine)

This function executes a command inside of the virtual environment. It can be used if you need to make changes to the virtual environment.

Parameters:

CommandLine

[in] CommandLine represents the application and commandline parameters to execute outside of the virtual environment

Returns:

Integer process ID, the process ID can be used with WaitForProcess ("WaitForProcess" on page 128)

Example:

```
ExecuteVirtualProcess("cmd.exe /c copy c:\systemfile.txt c:\virtualfile.txt")
```

Execute a command which requires quotes in the command-line

```
ExecuteVirtualProcess("regsvr32 /s " & chr(34) & "c:\program files\my.ocx" & chr(34))
```

Beta

GetBuildOption

Function GetBuildOption(OptionName)

This function returns the value of a setting specified in the [BuildOptions] section of the `Package.ini` file used for Thinstalling.

Parameters:

OptionName

[in] Name of the setting

Returns:

This function returns a string value. If the requested OptionName does not exist an empty string “” is returned.

Example:

```
Package.ini contains:
[BuildOptions]
DemoOption=DemoValue

Value = GetBuildOption("DemoOption")
```

Value = "DemoValue"

GetFileVersionValue

Function GetFileVersionValue(Filename, Value)

This function returns Version information value from a specific DLL, EXE, OCX, etc. This function can be used to determine the internal version number of a DLL or retrieve information about the DLL’s Copyright owner, or Product name associated with the DLL.

Parameters:

Filename

[in] The name of the filename where the version information is to be retrieved

Value

[in] The name of the Value to retrieve from the version information section of the specified file

The following values can be retrieved from most DLLs:

Comments
InternalName
ProductName
CompanyName
LegalCopyright
ProductVersion
FileDescription
LegalTrademarks
PrivateBuild
FileVersion
OriginalFilename
SpecialBuild

Returns:

This function returns a string value. If the requested Filename does not exist or the specified Value cannot be located in the file, the value of “” is returned.

Beta

Example:

```
FileVersion = GetFileVersionValue("c:\windows\system32\kernel32.dll", "FileVersion")

if FileVersion = "1.0.0.0" then
                MsgBox "This is Version 1.0!"

End if
```

GetCommandLine

Function GetCommandLine

This function allows you to access the command-line parameters passed to the current running program

Returns:

This function returns an a string that represents the command line arguments passed to the current running program, including the original exe.

Example:

```
MsgBox "The command line for this EXE was " + GetCommandLine
```

GetCurrentProcessName

Function GetCurrentProcessName

This function allows you to access the full path name of the current process.

Returns:

This function returns an a string that represents the full EXE pathname inside of the virtual environment. Typically this will be `c:\program files\...` even though the package source may be running from a network share.

Example:

```
MsgBox "Running EXE path is " + GetCurrentProcessName
```

GetOSVersion

Function GetOSVersion()

This function returns information about the current version of Windows

No Parameters

Returns:

This function returns a string in the following format:

`MAJOR.MINOR.BUILD_NUMBER.PLATFORM_ID OS_STRING`

`MAJOR` is one the following values

Beta

Windows Server 2003	5
Windows XP	5
Windows 2000	5
Windows NT 4.0	4
Windows Me	4
Windows 98	4
Windows 95	4
Windows NT 3.51	3

`MINOR` is one of the following values

Windows Server 2003	2
Windows XP	1
Windows 2000	0
Windows NT 4.0	0
Windows Me	90
Windows 98	10
Windows 95	0
Windows NT 3.51	51

`BUILD_NUMBER` is the build number of the OS

`Windows Me/98/95`: The low-order word contains the build number of the operatingsystem. The high-order word contains the major and minor version numbers.

PLATFORM_ID is one of the following values

`Value = 1` for Windows Me, Windows 98, or Windows 95 (Windows 95 based OS)

`Value = 2` for Windows Server 2003, Windows XP, Windows 2000, or Windows NT. (Windows NT based OS)

`OS_STRING` Represents additional information about the OS such as “Service Pack 2”

Example:

```
if GetOSVersion() = "5.1.0.2 Service Pack 2"
               then MsgBox "You are running on Windows XP Service Pack 2!"
endif
```

GetEnvironmentVariable

Function GetEnvironmentVariable(Name)

This function returns the environment variable associated with variable name Name.

Parameters:

Name

[in] The name of the environment variable for which the value is to be retrieved

Beta

Returns:

This function returns the string value associated with the environment variable “name”.

Example:

```
MsgBbox "The package source EXE is " + GetEnvironmentEnvironment("TS_ORIGIN")
```

RemoveSandboxOnExit

Sub RemoveSandboxOnExit(YesNo)

This function set toggles whether to delete the sandbox when the last child process exits.

If you set the package.ini option `RemoveSandboxOnExit=1`, the default clean up for the package with be “Yes”. In this case you, can change the clean up to “No” by calling RemoveSandboxOnExit with the value of 0. If you did not modify the `package.ini` option `RemoveSandboxOnExit=1`, the default clean up for the package with be “No”. In this case you, can change the clean up to “Yes” by calling RemoveSandboxOnExit with the value of 1.

Parameters:

YesNo

[in] Do you want to clean up when the last process shuts down? 1=Yes, 0=No

Example:

This will turn on cleanup

```
RemoveSandboxOnExit 1
```

This will turn off cleanup

```
RemoveSandboxOnExit 0
```

SetEnvironmentVariable

Sub SetEnvironmentVariable(Name, Value)

This function set the value of an environment variable.

Parameters:

Name

[in] The name of environment variable where the value is to be stored

Value

[in] The value to be stored

Example:

```
SetEnvironmentVariable("PATH", "C:\Windows\system32")
```

SetFileSystemIsolation

Sub SetFileSystemIsolation(Directory, IsolationMode)

This function sets the isolation mode (Chapter 4, “Isolation Modes,” on page 61) of a directory

Parameters:

Directory

[in] Full path of the directory whose isolation mode is to be set

Beta

IsolationMode

[in] Isolation mode to be set
1 = WriteCopy
2 = Merged
3 = Full

Example:

Set the isolation mode of the temp directory to Merged

```
SetFileSystemIsolation GetEnvironmentVariable("TEMP"), 2
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.215 or higher

SetRegistryIsolation

Sub SetRegistryIsolation(RegistryKey, IsolationMode)

This function sets the isolation mode of a registry key (Chapter 4, “Isolation Modes,” on page 61)

Parameters:

RegistryKey

[in] The registry key whose isolation mode is to be set. Start with “HKLM” for HKEY_LOCAL_MACHINE, “HKCU” for HKEY_CURRENT_USER of “HKCR” for HKEY_CLASSES_ROOT

IsolationMode

[in] Isolation mode to be set
1 = WriteCopy
2 = Merged
3 = Full

Example:

Set isolation of HKEY_CURRENT_USER\Software\Thinstall\Test to Full

```
SetRegistryIsolation "HKCU\Software\Thinstall\Test", 3
```

Availability:

This option requires VMware Project North Star (Thinstall) version 3.215 or higher

WaitForProcess

Function WaitForProcess(ProcessID, TimeOutInMilliSeconds)

This function waits until the specified ProcessID has completed execution.

Parameters:

ProcessID

[in] The processID to wait for completion. The process ID can come from ExecuteExternalProcess or ExecuteVirtualProcess

TimeOutInMilliSeconds

[in] The maximum amount of time to wait for the process to end before continuing. If 0 is specified, INFINITE is used

Beta

Returns:

This function returns an integer
0 = Timeout failed
1 = Process exited
2 = The process does not exists or security denied

Example:

```
id = ExecuteExternalProcess("cmd.exe")
WaitForProcess(id, 0)
```

Beta

Beta

Access Control using Active Directory

11

You can control access to application using Active Directory Groups. At build-time VMware Project North Star (Thinstall) will convert AD Group names into SID values. A SID is small binary value that unique identifies an object, similar to a GUID. SIDs are not unique for a few special groups like Administrator. Because SID values are stored in packages for later validation, this means a few different things:

1 You must be connected to your AD domain during build and the Groups you specified must exists. VMware Project North Star (Thinstall) will lookup the SID value during build.

2 If you delete a group and recreate it, the SID may change so you will need to rebuild the package in order to authenticate against the "new" group.

3 When users go offline, they can authenticate using cached credentials. Assuming a user can log into their laptop, VMware Project North Star (Thinstall) AD authentication will still work. You can control the period of time cached credentials are valid using Group Policy.

4 Cached credentials may not refresh on clients until the next AD refresh cycle. You can force a group policy on a client using the command "gpupdate". Sometimes the user may need to log-off before AD credentials are recached.

5 Special groups like Administrators and Everyone have the same SID on every Active Directory domain and Workgroup. Other groups you create will have a domain-specific SID, meaning a user cannot create their own local group with the same name to bypass authentication.

The option PermittedGroups=Group1;Group2

Example Package.ini files:

Example1

```
[BuildOptions]
PermittedGroups=Administrators;OfficeUsers
[App1.exe]
...
..

[App2.exe]
...
...
```

In this example App1 and App2 with both inherit PermittedGroups from "BuildOptions"

Example2

```
[BuildOptions]
PermittedGroups=Everyone

[App1.exe]
PermittedGroups=App1Users
AccessDeniedMsg=Sorry, you can't run this application
..
```

Beta

```
[App2.exe]
...
...
```

In this example, App1Users will be allowed to use `App1.exe` and Everyone will be allowed to use `App2.exe` The default message for denied user is also changed for App1

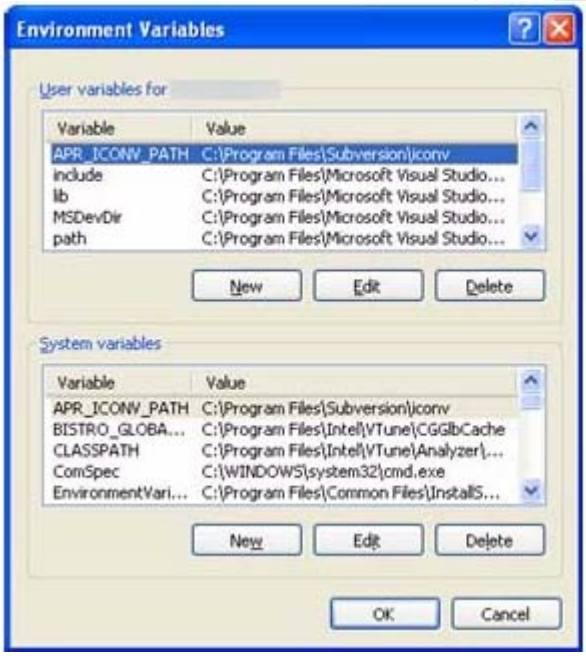
Beta

Application PATH and Environment Variables

12

SetupCapture will automatically capture system and user Environment Variables changes that occur indirectly by capturing and comparing the registry. If an application needs a specific environment variable in order to run correctly, you can make environment variable changes prior to completing the setup capture process. Typically the application's installer should do this automatically.

To manually add environment variables to VMware Project North Star (Thinstall) packages during the Setup Capture process, use **My Computer>Properties>Advanced>Environment Variables** You can add new variables to either System or User.



Windows stores environment variables on a system-wide and user-specific basis. The location for these two stores are located in the following locations.

System-wide environment variables:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment
```

User-specific environment variables:

```
HKEY_CURRENT_USER\Environment
```

For packages running under VMware Project North Star (Thinstall), any environment variables captured in these 2 locations will be applied for all applications running in the same Sandbox. Per-user values take precedence over per-machine values. You can manually edit `HKEY_LOCAL_MACHINE.txt` and `HKEY_CURRENT_USER.txt` to add environment variables after the capture has completed. For example, the following text can be appended to the end of `HKEY_LOCAL_MACHINE.txt` to add a CLASSPATH environment variable which contains the string `c:\mypath`

```
isolation_writecopy HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Environment
    Value=CLASSPATH
    REG_SZ~c:\mypath#00
```

NOTE the "Path" variable is ignored by VMware Project North Star (Thinstall), instead a new variable "VirtualPath" is used (see below).

The PATH variable

The PATH environment variable is treated differently because typically you do not want to completely replace the local system PATH variable, but instead prepend specific values. To accomplish this, SetupCapture uses a special value called "VirtualPath" to store the portions of the PATH variable that changed during setup capture. This VirtualPath will be prepended to the local system PATH during program execution.

Beta

Tips and Tricks 13

Visit our beta blog for more tips and tricks:

http://thinstall.com/blog

Use volume license keys where available, this usually eliminates problems where apps refuse to run on a machine different from the packaging machine

Applications that use with per-machine network activation may not work when run captured on one machine and run on a different machine.

When packaging applications that bind to specific host, don't run application prior to snap shot finalization—this will allow the application to activate on each individual desktop.

Some applications may store hard-coded path information in data files and may operate incorrectly when roaming to a new computer.

Beta

Packaging .NET Applications

Precaptured .NET

Because the capture process can be time consuming and requires several steps, we have made available pre-captured versions of the .NET Framework.

This simple guide is targeted towards software developers and IT Admins who want to deploy a .NET application without installing the .NET Framework.

What you can do with the pre-captured versions of the .NET Framework:

- You can build a `cmd.exe` application which is capable of running .NET applications from the host PC even though .NET may not be installed
- You can add your application files into the pre-captured .NET project and build a single EXE file which runs without installation of .NET

This guide should only be used for .NET applications that do not require installation of COM components or 3rd party components requiring registry entries.

- If you need to include Crystal Reports in your package, you should perform a full capture.
- If you need to include COM components in your package, you should perform a full capture.
- If you need to include customized registry settings, you should perform a full capture.

Express Setup (build virtual cmd.exe)

1 Download and Install Thinstall Virtualization Suite on any PC (signup at http://www.thinstall.com/products/virtualization_suite_dl.php)

2 Download .NET 1.1, 2.0, or 3.0 pre-captured VMware Project North Star (Thinstall) projects

You only need to download the version of the .NET Framework required to run your application

http://thinstall.com/examples/dnet11_captured.zip

http://thinstall.com/examples/dnet20_captured.zip

http://thinstall.com/examples/dnet30_captured.zip

3 Unzip your pre-captured VMware Project North Star (Thinstall) project from step 2

4 Click the `build.bat` file

5 In the `bin` directory, you will find `cmd.exe` you can now launch your .NET application using this `cmd.exe` like this:

```
cmd.exe /c myapplication.exe
```

You are done! Your application can now run on computers that don't have .NET

Professional Setup (build single EXE virtual app)

If you prefer to generate a single exe that runs without `cmd.exe`, follow these additional steps. First, let's assume you unzipped from Step #2 to `c:\project`, though this location doesn't matter.

1 Copy your .net application, dll files, and data files to `c:\project\%ProgramFilesDir%\myapp`

2 Edit `c:\project\package.ini`, change the line which says

From:

```
Source=%SystemSystem%\cmd.exe
```

To:

```
Source=%ProgramFilesDir%\myapp\myapp.exe
```

(where `myapp.exe` is your application's executable)

3 Run `c:\project\build.bat`

Also see: How to reduce your package size for .NET

Reducing size

Whether you download our pre-captured .NET Framework projects or you capture the .NET install yourself, the size of packages is often important. Here are a few tips for reducing your package size:

1 Turn on compression

The easiest way to generate smaller packages is to enable compression in your `package.ini` file. To do this, replace "Compression=None" with "Compression=Fast" in your `package.ini` file.

2 Delete any `.msi` files left behind by the installer

Many application leave behind their original installer in `.msi` format so that they can reinstall if they become corrupted, or if any features of the application were installed using "install on demand". You can safely delete the `.msi` files captured during the install of the .NET Framework.

Beta

3 Delete Native Images

.NET uses a process called Just in time compilation to convert IL byte code into native processor code at runtime. During installation, Native Image files will be generated to the hard drive. These files can increase your package size by 2 or 3 times, by deleting them, your packages can be much smaller. In .NET 1.1, this process was very fast and it could occur at startup time very quickly. In .NET 2.0 and higher, the native image compilation is significantly slower. There is a speed hit for startup time if you decide to delete native images because these images will be generated in memory rather than be pre-exiting in your VMware Project North Star (Thinstall)l package.

You can delete native images by deleting the directories:

```
%SystemRoot%\assembly\NativeImages*.*
```

Tips and Tricks for Specific Applications

The following sections describe tips and tricks that you can use to create packages for certain applications.

Office 2007

There is a potential conflict when you run a Thinstalled Office 2007 on a system which has a full native installation of Office 2003. In this configuration, the Thinstalled Office 2007 applications will detect that the "Handwriting" feature of Office 2003 was installed but, because of isolation, the applications won't see some of the files/registry values belonging to that feature. This will cause an automatic repair to be started, which shows up as a Windows Installer popup during each launch of a Thinstalled Office 2007 application. To prevent the repair, the files of the Office 2003 Handwriting feature need to be made visible to the Thinstalled Office 2007 apps. This can be done by changing the isolation mode on a couple of directories:

```
%ProgramFilesDir%\Common Files\Microsoft Shared\INK
%ProgramFilesDir%\Common Files\Microsoft Shared\Office11
%ProgramFilesDir%\Common Files\Microsoft Shared\Office11\1033
%ProgramFilesDir%\Microsoft Office
```

(Depending on the language version of your installation, you might have a different directory instead of `%ProgramFilesDir%\Common Files\Microsoft Shared\Office11\1033`.)

After a fresh capture, these directories will contain a `##Attributes.ini` file ("##Attributes.ini" on page 145) with "`DirectoryIsolationMode=Full`". Change this to "`DirectoryIsolationMode=WriteCopy`". The isolation mode of a couple of registry files needs to be adjusted too. Edit `HKEY_LOCAL_MACHINE.txt` and search for these keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Components\379E92CC2CB71D119A12000A9CE1A22A
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Components\1650EACF3C291D11A92C0006794C4E25
```

Change the "`isolationmode_full`" at the beginning of the lines to "`isolationmode_writecopy`".

The Outlook Tips and Tricks ("Outlook" on page 137) also apply to Outlook 2007.

Outlook

Outlook stores its account settings etc. in a number of registry keys and files. When you start up Outlook for the first time, it will check if these keys exist and, if not, will prompt the user to create a new account. This works fine in the virtual environment when Outlook is not installed on the physical system. However, when the user already has Outlook installed physically, the Thinstalled version will find the registry keys in the system registry and use those settings, which is probably not what's intended. What's needed is "Full Isolation Mode" (Chapter 4, "Isolation Modes," on page 61) for the registry keys and files where Outlook stores its settings, so the Thinstalled version will pretend the system systems don't exist. To set this up, some changes need to be made to the capture. First of all, the following entries need to be added to the `HKEY_CURRENT_USER.txt` file:

Beta

```
isolation_full HKEY_CURRENT_USER\Identities
isolation_full HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Windows Messaging Subsystem\Profiles
```

Next, a file named `##Attributes.ini` ("##Attributes.ini" on page 145) with the following contents:

```
[Isolation]
DirectoryIsolationMode=Full
```

must be created in each of the following subdirectories:

```
%AppData%\Microsoft\AddIns
%AppData%\Microsoft\Office
%AppData%\Microsoft\Outlook
%Local AppData%\Microsoft\FORMS
%Local AppData%\Microsoft\Outlook
```

(create the subdirectories as needed)

Attachments

By default, Outlook creates a directory where it stores attachments when you open an attachment for viewing. This directory will normally be something like `C:\Documents and Settings\<username>\Local Settings\Temp\Temporary Internet Files\OLKxxxx` where the last "xxxx" is replaced by something randomly chosen. This works fine when the viewing application runs in the same virtual sandbox as Outlook. However, when you want to use an external application (an application not running in Outlooks virtual sandbox) the external application won't be able to find the file it is supposed to show. This is because Outlook will store the file in the sandbox. To solve this, the isolation mode of the directory where the attachments are stored needs to be set to Merged. The next problem then is that we don't know what the name of that directory is (the last part of the name was randomly chosen). To solve this, add a value to `HKEY_CURRENT_USER.txt` which sets the name of attachment directory:

```
isolation_full
HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Outlook\Security
               Value=OutlookSecureTempFolder
               REG_SZ~%Profile%\Local Settings\OutlookTempxxxx#2300
```

(The "11.0" in the key name is for Outlook 2003, use the appropriate version number for your version of Outlook). For slightly increased security, replace the last 4 "xxxx" characters by something random. Next, create a directory `%Profile%\Local Settings\OutlookTempxxxx` in your project and create an `##Attributes.ini` file ("##Attributes.ini" on page 145) there containing:

```
[Isolation]
DirectoryIsolationMode=Merged
```

The directory `%Profile%\Local Settings\OutlookTempxxxx` is just an example, you can use whatever you want, as long as you make sure that the directory is named in the OutlookSecureTempFolder registry key and is set to the correct isolation mode.

Explore.exe

The Windows Shell (`explorer.exe`) will check for an already running copy using `GetShellWindow()`. If `explorer.exe` detects a shell is already running, it will send a message to the running copy and exit. If you want to run `explorer.exe` in the virtual environment, you can do the following steps:

1 Kill the currently running process `explorer.exe` using Task Manager

2 Run the `explorer.exe` from a VMware Project North Star (Thinstall) virtual environment

Beta

Java Runtime Environment (JRE)

A conflict may occur if one version of Java is installed natively and another one is included in a Thinstalled executable. This is because newer versions of Java also install a plugin DLL that gets loaded by IE. This plugin DLL overwrites virtual registry keys when it is loaded and can conflict with a virtualized copy of older Java runtimes. You can prevent IE from loading plugins by adding the following line to beginning of `HKEY_LOCAL_MACHINE.txt`

```
isolation_full HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects
```

Beta

Beta

Snapshot.exe

14

Typically you do not need to use `snapshot.exe` directly, it is invoked by SetupCapture. This documentation is for advanced users and system integrators who are building VMware Project North Star (Thinstall) functionality into other platforms.

Command Line Usage:

NOTE all parameters are case-insensitive

```
snapshot CaptureFile.snapshot
[BaseDirectory1][BaseDirectory2][BaseRegistry1][BaseRegistry2][–Config ConfigFile.ini]
snapshot Original.snapshot –Diff NewEnvironment.snapshot OutputDirectory [–Config
ConfigFile.ini][–Quiet]
snapshot Original.snapshot –DiffPrint NewEnvironment.snapshot [–Config ConfigFile.ini]
```

snapshot can be used for 2 purposes, saving a snapshot and creating a VMware Project North Star (Thinstall) project from 2 previously captured snapshots.

Saving a snapshot

Snapshot can capture a snapshot of a computer's filesystem and registry which is then used later to create a VMware Project North Star (Thinstall) project. If no additional parameters, Snapshot will scan and save a copy of:

- All file information for all local drives (Directories, Filenames, File Attributes, File Sizes, and File modification dates)
- A complete copy of the registry trees `HKEY_LOCAL_MACHINE` and `HKEY_USERS` (because HKCR and HKCU are subsets of HKLM and HKU, there is no need to scan them)

Example usage:

```
snapshot c:\Capture.snapshot
```

Captures a complete snapshot of local drives and registry to the file `c:\Capture.snapshot`

```
snapshot c:\Capture.snapshot c:\ e:\
```

Captures a complete snapshot of the drives `c:\` and `e:\`, no registry information is captured

```
snapshot c:\Capture.snapshot c:\ HKEY_LOCAL_MACHINE\Software\Classes
```

Captures a complete snapshot of the drive `c:\` and all of the `HKEY_CLASSES_ROOT` registry subtree

Creating a VMware Project North Star (Thinstall) project from 2 previously captured snapshots.

Snapshot can be used to generate a VMware Project North Star (Thinstall) project directory by comparing 2 snapshots.

Example usage:

```
snapshot c:\Original.snapshot –Diff c:\NewEnvironment.snapshot c:\MyProject
```

Beta

Compares the two specified projects, and generates a resulting VMware Project North Star (Thinstall) Project in the directory `c:\MyProject`

Displaying Differences between 2 previously captured snapshots

```
snapshot Original.snapshot –DiffPrint NewEnvironment.snapshot
```

DiffPrint is similar to -Diff except that is does not have an output directory and all changes detected are printed to the console

Config Files

The Config file allows you to specify directories and subkeys to be excluded from a scan of created projects.

If no config file is specified Snapshot will attempt to load it's config file from the locations:

1 `Application Data\Thinstall\Spanshot.ini` (User's `AppData` directory)

2 `C:\Program Files\Thinstall\Snapshot.ini` (Location where `Snapshot.exe` is run from)

See the default `Snapshot.ini` in the `C:\Program Files\Thinstall\Snapshot.ini` for more details about customizations that can be performed.

Beta

Virtual Registry

15

VMware Project North Star (Thinstall) Registry information exists in 3 forms:

- Build format—This is unicode text files such as `HKEY_LOCAL_MACHINE.txt`, the build process converts this format into the embedded format.
- Embedded format (read-only)—This is compressed and stored inside the application `EXE` file in a binary format.
- Sandbox format (read-write)—This stores the differences from the embedded format as the application makes registry writes

Packaged/Compiled Format

After you build a package, all registry values and files will exist in a read-only image inside of the EXE you are distributing. At runtime the read-only image presents a view of the registry for new users of a package. As the application modifies the registry, these changes will be saved in the sandbox persistent file with the extension .tvr. The VRegTool utility ("VREGTool" on page 89) can be used to list and modify data contained in a `.tvr` file.

Build Format

This is what registry data looks like after you perform setup capture. In this format, each registry subkey will map directly to a directory on your build filesystem. Each registry value also maps directly to a file. An example view looks like this:

```
c:\mypackage\HKEY_LOCAL_MACHINE\Software\ApplicationX\Value1.txt
c:\mypackage\HKEY_CLASS_ROOT\.txt
```

Registry subkey & value name -> Filename mapping

Because registry subkeys and value names can have some characters which do not map to filenames, any unmappable character must be escaped in the filename like this:

Registry Subkey: `HKEY_CLASS_ROOT\*.*`

Maps to filename: `c:\mypackage\HKEY_CLASS_ROOT\#42.#42`

Because the character '*' is not a valid filename character, it has been escaped as #42

Because '#' is used to escape characters, it must also be escaped.

To represent a default value, the filename ##default is used.

NOTE FAT32 versus NTFS

FAT32 has severe limits on maximum filename lengths, it is highly recommend that you store all build registry data either on a NTFS drive or use a short parent directory name to store registry build data.

At runtime FAT32 limitations will not impact the registry.

Beta

Registry Value data

Registry files are saved as plain txt or Unicode text. Unicode text files start with the two byte sequence hex(ff),hex(fe). Notepad and Wordpad will automatically text unicode/ansi text files.

```
TYPE=VALUE
```

TYPE can be one of the following:

```
REG_NONE, REG_SZ, REG_EXPAND_SZ, REG_BINARY, REG_DWORD,
REG_DWORD_LITTLE_ENDIAN, REG_DWORD_BIG_ENDIAN, REG_MULTI_SZ,
REG_RESOURCE_LIST, REG_FULL_RESOURCE_DESCRIPTOR,
REG_RESOURCE_REQUIREMENTS_LIST
```

If type is string data (REG_SZ, REG_EXPAND_SZ, or REG_MULTI_SZ), then VALUE will be listed as sequence of characters terminated by a newline or carriage return character. Any unprintable character should be escaped into 2 hex characters:

```
REG_SZ=This is a line#0awith a carriage return as part of the data#00
REG_MULTI_SZ=This is a line1#00This is line2#00#00
```

Additionally escaped characters

because the '#' represent an escape character, all '#' characters must also be escaped. the '%' is used for VMware Project North Star (Thinstall) macro expansion, it must also be escaped when it the value should remain unexpanded.

Example Macro expansion:

```
REG_SZ=%ProgramFilesDir%\ApplicationX\control.ocx
```

at runtime, when the program accesses this registry value, it will be expanded by VMware Project North Star (Thinstall)

Virtual Registry—Text Format

Text format for the virtual registry is similar to regedit .reg format, however it supports macro expansion for string registry values names and value data.

The format looks like:

```
isolation_mode FULL_SUBKEY_NAME

               Value=VALUE_NAME
               VALUE_TYPE=VALUE_DATA
```

OR

```
               Value=VALUE_NAME
               VALUE_TYPE~MACRO_VALUE_DATA
```

`isolation_mode`—Can be one of VMware Project North Star (Thinstall)'s isolation modes (Chapter 4, "Isolation Modes," on page 61), as specified by "isolation_full", "isolation_merged", or "isolation_writecopy".

`VALUE_TYPE`—Specifies one the value data registry type. This should be one of the following:

```
REG_SZ, REG_EXPAND_SZ, REG_BINARY, REG_DWORD, REG_DWORD_BIG_ENDIAN,
REG_LINK, REG_MULTI_SZ,
REG_RESOURCE_LIST, REG_FULL_RESOURCE_DESCRIPTOR,
REG_RESOURCE_REQUIREMENTS_LIST
```

If you need to represent a non-published registry type, alternatively you can specify a decimal (base 10) number. For example:

```
453=#00
```

`VALUE_DATA`—Specifies value data. Data is represented differently depending on if the data type is a string or non-string type.

Beta

String Types (REG_SZ, REG_MULTI_SZ, REG_EXPAND_SZ)—Values are represent as escaped string values. A Unicode text file must be used to represent non-ansi characters. VMware Project North Star (Thinstall) uses unicode text files by default. The '#', newline character, tab character, carriage return, end-of-file, and NULL characters must be escaped if they exist in the original string.

Escaped characters begin with '#' and are followed by 2 hex characters to represent the ANSI value for the character. Non-ansi Unicode characters do not need to be escaped but can only exist in unicode text files.

Examples:

REG_SZ=**Both#00**

This represents a 5 character string value, the 5th character being a NULL (0) character.

REG_MULIT_SZ=**String1#00String2#00#00**

This represents a multi-string value which contains a list of 2 values "String1" and "String2"

REG_DWORD=**#27#c6#00#02**

This represents a 4 byte binary value.

For binary values, data is stored as a string of escaped bytes. For DWORD values, data is stored in native intel order (little endian) so no special processing needs to be performed for binary data.

MACRO_VALUE_DATA—Specifies value data which should be macro-expanded prior to use by the application.

For example, the value %AppData% will be expanded at runtime to the location of the user's Application Data directory.

If value data is not macro-expanded, the application would receive the literal string "%AppData%" when queries this registry value.

Macro-expanded data is only supported for string value types (REG_SZ, REG_EXPAND_SZ, and REG_MULTI_SZ)

Examples:

REG_SZ~**%AppData%**

Represents the value of the "Application Data" folder on the host PC where VMware Project North Star (Thinstall) is running.

The length of this registry value will change depending on what machine it runs on.

REG_SZ~**#23AppData#23**

Represents the value of the actual value "%AppData%". This value will always be the literal string "%AppData%" regardless of the current shell folder location.

```
isolation_full HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{047a9a40-657e-11d3-8d5b-
               00104b35e7ef}\InprocServer32
Value=ThreadingModel
REG_SZ=Both#00
Value=
REG_SZ~%SystemSystem%\mscoree.dll#00
```

##Attributes.ini

This file controls per registry subkey settings.

RegistryIsolationMode=WriteCopy | Merged | Full (Chapter 4, "Isolation Modes," on page 61)

example:

RegistryIsolation=WriteCopy

If isolation is not specified, it will be inherited from parent subkey.

Beta

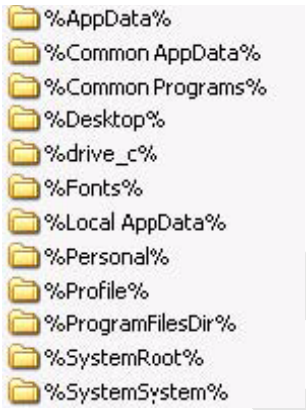
Beta

Virtual Filesystem 16

VMware Project North Star (Thinstall) virtual filesystem is represented in 3 different stages:

1 The package build format. This format stores files directly on the normal file system as they are to be compiled into the embedded virtual file system. Folder Macros ("Folder Macros" on page 148) are used to reprent Windows standard shell folder locations.

   A typical view of the virtual files in pre-build package format:



2 The embedded virtual filesystem (read-only). This filesystem is embedded in EXE files during the build process. The read-only filesystem may be compressed and provides block-based streaming ("Block based Streaming" on page 53) to client PCs.

3 The file system sandbox (read-write). This directory structure holds file data which was modified by the application. Any file modification operation will cause embedded virtual files to be extracted to the sandbox, including:

   - Changing a file's timestamp or attributes
   - Opening file with write access
   - Truncating file
   - Renaming / Moving file

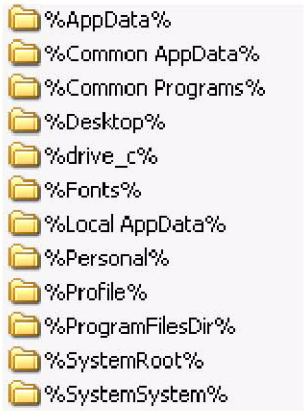
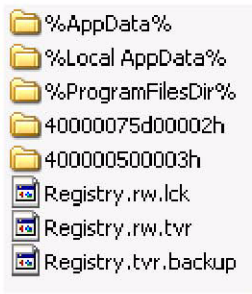
Prior to building an application, you can adjust isolation modes (Chapter 4, "Isolation Modes," on page 61) on a per-directory basis using `##Attributes.ini` files ("##Attributes.ini" on page 150). Isolation modes enable you to specify whether the application is presented with a merged view of the virtual file system or only see virtual files. Isolation modes also control what happens when an application tries to write to a directory.

Both the embedded and sandbox file systems uses Folder Macros ("Folder Macros" on page 148) to enable file paths to dynamically expand at runtime.

Beta

Folder Macros

VMware Project North Star (Thinstall) uses macros to represent file system path locations that may relocate when running on different OSes or PCs. The table below list the available macros. When creating a new VMware Project North Star (Thinstall) project using Setup Capture, these macros will automatically hard-coded paths with macro forms. A typical project root directory will look something like this:

Typical project structure (used to build EXE file)	Typical Sandbox Directory (holds runtime modifications)
	

Beta

Macro Name	Shfolder.dll ID	*Ver	Typical location
%ProgramFilesDir%	CSIDL_PROGRAM_FILES	5.0	C:\Program Files
%Common StartMenu%	CSIDL_COMMON_STARTMENU		C:\Documents and Settings\All Users\Start Menu
%Common AppData%	CSIDL_COMMON_APPDATA	5.0	C:\Documents and Settings\All Users\Application Data
%Local AppData%	CSIDL_LOCAL_APPDATA	5.0	C:\Documents and Settings\username\Local Settings\Application Data
%AppData%	CSIDL_APPDATA	4.71	C:\Documents and Settings\username\Application Data
%Common Desktop%	CSIDL_COMMON_DESKTOPDIRECTORY		C:\Documents and Settings\All Users\Desktop
%Desktop%	CSIDL_DESKTOPDIRECTORY		C:\Documents and Settings\username\Desktop
%Common Startup%	CSIDL_COMMON_STARTUP		C:\Documents and Settings\All Users\Start Menu\Programs\Startup
%Startup%	CSIDL_STARTUP		C:\Documents and Settings\username\Start Menu\Programs\Startup
%Common Programs%	CSIDL_COMMON_PROGRAMS		C:\Documents and Settings\All Users\Start Menu\Programs
%Programs%	CSIDL_PROGRAMS		C:\Documents and Settings\username\Start Menu\Programs
%Common Favorites%	CSIDL_COMMON_FAVORITES		C:\Documents and Settings\All Users\Favorites
%Favorites%	CSIDL_FAVORITES		C:\Documents and Settings\username\Favorites
%SendTo%	CSIDL_SENDTO		C:\Documents and Settings\username\SendTo
%Templates%	CSIDL_TEMPLATES		C:\Documents and Settings\username\Templates
%Recent%	CSIDL_RECENT		C:\Documents and Settings\username\My Recent Documents
%NetHood%	CSIDL_NETHOOD		C:\Documents and Settings\username\NetHood
%Fonts%	CSIDL_FONTS		C:\Windows\Fonts
%My Pictures%	CSIDL_MYPICTURES	5.0	C:\Documents and Settings\username\My Documents\My Pictures
%My Videos%	CSIDL_MYVIDEO	6.0	C:\Documents and Settings\username\My Documents\My Videos
%Personal%	CSIDL_PERSONAL	6.0	C:\Documents and Settings\username\My Documents
%Profile%	CSIDL_PROFILE	5.0	C:\Documents and Settings\username
%Profiles%	CSIDL_PROFILES	6.0	C:\Documents and Settings

*Ver: VMware Project North Star (Thinstall) uses `shfolder.dll` to obtain the location of Shell folders. Older versions of `shfolder.dll` do not support some Macro Names, if there is a requirement for a specific version of `shfolder.dll` on the host OS, this has been listed.

*** Special processing for %SystemRoot% When running in a Terminal Services environment, there are 2 “Windows” directories, a shared Windows directory (e.g. `C:\Windows`) and a private Windows directory (e.g. `C:\Documents and Settings\User\Windows`). For Terminal Services environments, VMware Project North Star (Thinstall) will use the user-specific directory for `%Systemroot%`

##Attributes.ini

This file controls per directory settings.

```
[Isolation]
DirectoryIsolationMode=WriteCopy | Merged | Full (Chapter 4, “Isolation Modes,” on page 61)
```

example:

```
DirectoryIsolation=WriteCopy
```

If isolation is not specified, it will be inherited from parent directory.

```
[Compression]
CompressionType=None | Fast | Small
BlockSize=64k | 128k | 256k | 512K | 1M
BlockType=Fixed | Rolling

[FileList]
ExcludePattern=*.tmp,*.obj,FullName
```

Beta

Frequently Asked Questions

17

Q. When I complete the capture process and build, why are there are no EXE files generated?

A. Setup Capture uses shortcuts installed during installation to determine which EXEs to build ("Package.ini generation" on page 72). The most common reason for this problem occurs if you instruct the installer not to write any shortcuts during capture, or in some case applications specifically do not install any shortcuts, for these applications you need to manually edit the `package.ini` file ("Package.ini generation" on page 72).

Q: Can VMware Project North Star (Thinstall) applications interact with other applications, printers, drivers, etc.?

A: Yes, for the most part there is no difference between a Thinstalled application and system installed application. See "Desktop Integration" on page 56 for more details.

Q: What is the performance overhead running when a Thinstalled application?

A: VMware Project North Star (Thinstall) is an application level Virtualization solution. The total disk footprint for deploying VMware Project North Star (Thinstall) with an application is approximately 400kb versus 4 Gigabytes needed to deploy a virtual machine. VMware Project North Star (Thinstall) does not emulate or translate any application instructions, so they run at full native speed. The memory impact from running a Thinstalled application will vary depending on how many DLLs are loaded, but in general you should expect the application to require approximately 2MB of RAM over the original requirements.

Q. Does the size of the package effect how quickly it starts up or how much memory it consumes?

A. No, VMware Project North Star (Thinstall) uses block-based streaming ("Block based Streaming" on page 53) to load only the required parts of an application into memory as needed. A two gigabyte package can start executing in less than a second and consume less than 10MB of RAM. The size of the package has no correlation to the amount of memory or startup time required.

Q: How do I control application use and maintain license compliance?

A: You can currently control access to Thinstalled package through a few different mechanisms:

- Active Directory (Chapter 11, "Access Control using Active Directory," on page 131): In the `package.ini` file, you can specify which users groups are able run a specific package. Then using AD you can centrally add and remove people to groups as needed.
- VB Scripts (Chapter 9, "Scripting," on page 117): This allows you to "roll-your-own" system. VB Script gives you a lot of power, and you can easily build hard-coded package times with a simple script. We have posted a simple sample VB script which will expire packages on a specific date, you could also easily check for a specific machine registry key before allowing the user to run the application. http://thinstall.com/thintalk/viewtopic.php?t=205

  VB Script can easily call any COM component that is registered in the virtual environment, so you can also write a C++ DLL that is called and executed by vb script before the program starts and after it shuts down.

We will also have a webserver based license management system available in April 2007:

Beta

http://thinstall.com/thintalk/viewtopic.php?t=207

Q. Can VMware Project North Star (Thinstall) packages be signed?

A. Yes, Thinstalled EXE files can be signed using standard Microsoft Internet Digital Signatures (Authenticode http://www.microsoft.com/technet/archive/security/topics/secaps/authcode.mspx?mfr=true). For Software publishers, signed executable gives end-users assurance that the executable has not been modified after it was signed. System Administrators can also use the Internet Explorer Administration Kit http://technet.microsoft.com/en-us/ie/bb219517.aspx (IEAK) Configuration Wizard to prevent user groups from downloading/running unsigned code.

Q: How do I apply patches and updates to an application?

See "Application Updates" on page 153.

Q: How do I create a file type association and shortcut?

A: Using the ThinReg utility ("ThinReg" on page 86)

Q: Do Thinstalled applications work on my Citrix server?

A: Yes, VMware Project North Star (Thinstall) runs on Citrix and Terminal Server exactly the same single-user Desktop environments.

Q: What does the sandbox do and how do I control the sandbox location?

A: See the help topic Chapter 5, "Sandbox Overview," on page 65

Q: What are isolation modes and how do I control them?

A. Isolation modes allow you to control what an application can read and write on the local PC. Read isolation can allow system installed and virtual VMware Project North Star (Thinstall) packages to run simultaneously without conflicting with each other. Write isolation allows VMware Project North Star (Thinstall) packages to believe they have global write permissions while they really only modify the sandbox directory.

More information about isolation modes can be found here. ("Understanding Isolation Modes" on page 62)

Q. Does VMware Project North Star (Thinstall) decompress package data to disk when it runs the first time? How does compression work and what is the penalty?

VMware Project North Star (Thinstall) uses block based streaming ("Block based Streaming" on page 53) to decompress file data from disk directly into memory a block at a time as requested by the application. Because data is not written to disk and decompression is very efficient, the performance impact for using compression is usually negligible. When loading applications from slower media such as USB flash or network shares, compression helps to accelerate loading times because less data needs to be transferred. Because data is not cached or decompressed to disk first, no extra disk space is required before, during, or after executing Thinstalled applications.

Q: Does VMware Project North Star (Thinstall) work with .Net applications?

A: Yes—if you are deploying an application that requires .NET, there a number of different options available. You can include .NET in your package by capturing on a clean machine which doesn't have .NET already installed (such as clean Win2k or XP SP1). If you start with a clean machine, install .NET first and then do the capture process the resulting application will use .NET from the host PC instead of being included in the package. If you don't want to capture the .NET Framework yourself, we have some pre-captured projects you can use.

https://thinstall.com/thintalk/viewtopic.php?t=203

Q: Can I VMware Project North Star (Thinstall) application using Java or ActiveX components in a browser?

A: These flash demo shows how you can virtualize Internet Explorer plugins/addins such as the Java Runtime Environment using VMware Project North Star (Thinstall) VS.

http://thinstall.com/demos/java_applet/

Beta

http://thinstall.com/demos/ica_activex/

Q: Does VMware Project North Star (Thinstall) support applications using services?

A: Yes, VMware Project North Star (Thinstall) supports virtual services. If a captured package installs a service, VMware Project North Star (Thinstall) will automatically start the service as virtual service in the same user account as the logged in user as needed by the application.

Q. Does VMware Project North Star (Thinstall) support plugins and Addins?

A: Yes, There are a number of different ways plugins can be packaged together or separate from applications. ("Plugins and Addins" on page 154)

Q: Can VMware Project North Star (Thinstall) run on XPe (embedded)?

A: Yes.

Q: Is there a way just to re-run the post-scan before doing the build?

I did the pre-scan, installed the app and then ran the post-scan. After doing the post-scan I realized that I wanted to change the configuration of the app slightly and re-run the post-scan but I can't seem to find away to just run the post-scan again on it's own.

A: Yes, if you premature clicked the "Post-install scan", you can just hit the "back>>" button and it will take you to the post-install scan step again and continue from there. The back button also always to create multiple similar VMware Project North Star (Thinstall) projects in the same session.

Alternatively you can also create projects from the command-line using the snapshot tool like this:

1 `snapshot.exe c:\snap1.dat`

2 (install application)

3 `snapshot.exe c:\snap2.dat`

4 `snapshot.exe c:\snap1.dat –diff c:\snap2.dat c:\myproject`

Q: Why can't I delete or rebuild my application, the file appears to be locked

A: Thinstalled EXE files and some associated sandbox files will be locked while the virtualized application is running and cannot be deleted or altered. This also applied for any virtual child process started by the parent process. Sometime application will purposefully leave processes behind after they quit, for example Office 2003 will start a process called `ctfmon.exe` and leave it running even after Word or Excel have quit. You need to manually kill these child processes before you can rebuild the application. Luckily VMware Project North Star (Thinstall) gives you a tool to hunt down any Thinstalled processes that might be running. This tool is "`dll_dump.exe`" can be found in `c:\program files\thinstall.vs` and run likes this "`dll_dump *`". If the application is locked on a user's desktop, VMware Project North Star (Thinstall) has the ability to perform in-place updates ("Upgrading applications" on page 57).

Q: Why don't icons display in the Windows Shell for large EXE files (>1GB)?

A. This is due to limitations in how Windows is implemented, a work-around is available ("Large Packages" on page 155).

Application Updates

What happens if an application has auto-updating capabilities?

If an application has built-in auto updating capabilities, then it's update mechanism will still function when running under VMware Project North Star (Thinstall). If the application downloads and runs an installer/patching program, this will run inside of the virtual environment and all of the changes made by the update software will be written to the sandbox. When the application restarts it will use the version of the EXE in the sandbox, not from the original package.

Beta

For example if you package Firefox 1.5 and run it, it will ask if you want to upgrade to Firefox 2.0 – if you click yes, it will download updates and write them to the sandbox and prompt you to restart. If you run the application it again, you will be running Firefox 2.0. By deleting the sandbox it will revert back to version 1.5.

Thinstall's Sandbox Merge utility ("sbmerge" on page 85) can be used to merge changes made by an application's update mechanism back into the original package structure in order to build an EXE to redeploy to end-user machines.

VMware Project North Star (Thinstall) opens new possibilities for updating applications dynamically without requiring Administrator rights. For example .NET based applications that download new DLL files from the internet as part of their update process need to execute `ngen.exe` to generate Native Image assemblies in order to improve startup performance. Normally `ngen.exe` writes to `HKLM` and `c:\windows`, both of which are off-limits for non-Admin accounts. When running under Thinstall `ngen.exe` can install Native Image assemblies even on Guest user accounts, however these changes will be stored in a user-specific directory.

How can I update applications from a central location?

You may want to update the package on a central computer and push the changes to clients or central network shares as new Thinstalled `EXE` file.

The first step is to create a VMware Project North Star (Thinstall) project which contains the application updates. There are three main approaches for accomplishing this:

Approach #1: Apply patches during capture. This approach is very simple and fairly error-proof.

Approach #2: Apply patches inside the virtual environment. If any application offers auto-update capabilities, it can update itself as described above. If the application update is in the form of a `patch.exe`, the patch program can be run in the virtual environment creating a `cmd.exe` entry point and running the patch program from there. The changes that occur during auto-update or manual update will go into the sandbox so you can easily roll-back to the original version by deleting the sandbox. You can apply patches in the virtual environment either on an end-user desktop machine or on a central packaging machine. In the second case, sandbox changes made by the patch or update can be merged back into the application using Thinstall's Sandbox merge utility.

Approach #3: Apply patches to already captured project. If you only need to update a small set of specific files or registry keys, you can simply replace the files in your captured project. This approach is well suited for software developers who can integrate VMware Project North Star (Thinstall) builds into their normal "make" process.

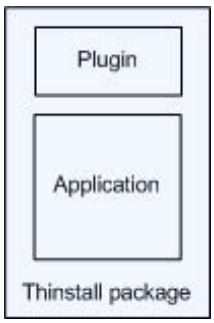
Beta

Plugins and Addins

Many application have the ability to install plugins or load external components. VMware Project North Star (Thinstall) provides flexibility on plugins can be packaged together or separately from an application.

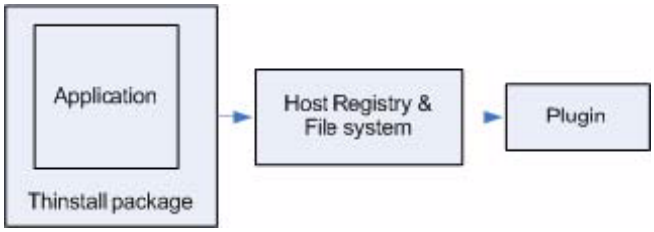
Scenario: I want to deploy a virtualized application plus Add-Ins in a single package

This scenario is the simplest. To create a single Thinstalled application which contains both the main application and it's Add-in's, simply install the add-in's at the same time as the application during the capture process. The virtualized application will be able to load and use the add-in's as if they were installed on the host PC.



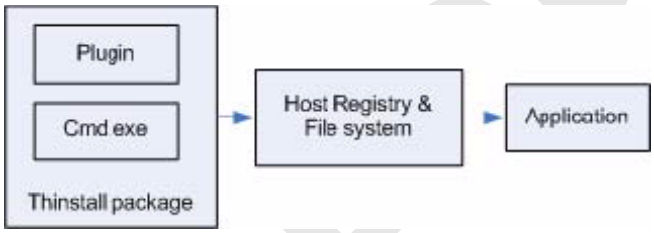
Scenario: I want to have a virtualized application load plugins from the host PC

This scenario may require specific knowledge on how the application works and tweaks to the package to allow the virtualized application to see plugins installed on the host PC. For example, it is common for applications to load plugins based on specific registry values or file system locations. For default projects created by setup capture, some registry subtrees and directories will be setup with full isolation (Chapter 4, "Isolation Modes," on page 61). Full Isolation prevents the application from seeing host PC registry keys or files so the application may not be see plugins that exist on the host PC. If you are not sure which registry subtrees or directories are used to locate and load plugins, VMware Project North Star (Thinstall)'s Log Monitor ("Log Monitor" on page 74) can help you track down that information. By changing the isolation mode from full to writecopy or merged, you enable the application to see and load plugins from the host PC.



Scenario: I want a system-installed version of an application to load plugins from a Thinstalled package without installation.

This scenario is can be achieved by capturing the install of a plugin by itself and building a `cmd.exe` package. The virtualized `cmd.exe` can then be used to launch a system-installed application. The system installed application will run inside of the virtualized environment and see and load your virtualized plugins. You can also launch the system installed application automatically by running "`cmd.exe /c c:\myapp.exe`"



Large Packages

Windows Shell Limitations for displaying Icons

VMware Project North Star (Thinstall) supports packages of unlimited sizes, however on some Windows platforms the windows shell (`explorer.exe`) has limitations on how it displays icons and may not display the icon for an EXE file once it reaches a certain size.

`Explorer.exe` tries to map the entire EXE file into memory so it can access the Icon resources. If there is not enough available virtual memory address space, this file mapping operation will fail. Typically an application will have a maximum of 2GB of address space, though DLLs, the process heap, and other objects will break up the virtual address space so a smaller maximum block is actually available. On most computers, Windows XP will not display icons for EXE files larger then 600MB. Windows Vista does not appear to have this limitation.

Luckily there is a very easy work-around for this problem. When building a large package, you can create two files:

- One large file containing the package data
- One small file containing Icon Information

Example (Before):

package.ini

```
------------------------------------------------
[LargeApplication.exe]
Source=%ProgramFilesDir%\LargeApplication.exe
ReadOnlyData=bin\Package.ro.tvr
------------------------------------------------
```

Example (After):

package.ini

```
------------------------------------------------
[LargeApplication.dat]
Source=%ProgramFilesDir%\LargeApplication.exe
ReadOnlyData=bin\Package.ro.tvr

[LargeApplication.exe]
Source=%ProgramFilesDir%\LargeApplication.exe
Shortcut=LargeApplication.dat
------------------------------------------------
```

Beta

Index

Beta

Beta